Fast Text Mining Using Kernels in R

Ingo Feinerer[1] and Alexandros Karatzoglou[2]

[1] Theory and Logic Group
Institute of Computer Languages
Vienna University of Technology
Austria, *feinerer@logic.at*

[2] LITIS, INSA de Rouen
Avenue de Universite
76801 Saint-Etienne du Rouvray
France, *alexis@ci.tuwien.ac.at*

Abstract. Recent advances in the field of kernel-based machine learning methods enable the fast processing of text using string kernels which are built with the use of suffix arrays. `kernlab` provides both kernel methods infrastructure and a large collection of already implemented algorithms and includes an implementation of suffix array based string kernels. Along with the use of `tm` these packages provide `R` with functionality in processing, visualizing and grouping large collections of text data using kernel methods. We focus on the performance of various types of string kernels at these tasks.

Keywords: string kernels, text mining, clustering, R

1 Introduction

Kernel-based methods are state of the art machine learning methods who have gained prominence mainly through the successful application of Support Vector Machines (SVMs) in a large domain of applications. SVMs have also been applied in classification of text documents using typically the inner product between two vector representations of text documents.

String kernels (Watkins (2000), Herbrich (2002)) have proven to be a promising alternative providing excellent results in text classification using SVMs and clustering using kernel based clustering methods like e.g. spectral clustering. One of the main drawbacks in the application of string kernels in large document collections is that until recently most algorithms and implementations where both slow and scaled usually in $O((n+m)^2)$ where n and m are the number of characters in the two documents. Vishwanathan and Smola (2004) introduced string kernels based on suffix trees which scaled $O(n+m)$ but had still drawbacks since the construction and use of a suffix tree requires a large amount of memory (typically $40n$) and has poor locality. Suffix trees based string kernels scale in theory in $O(n)$ but have been proven to be relatively slow for large scale text processing due to this inherent weakness. The use of suffix arrays (Teo and Vishwanathan (2006)) resolved most of these issues and has transformed string kernels into a very useful option for large scale text mining using kernel-based machine learning.

In this paper we will demonstrate the viability of kernel-based machine learning for text mining using suffix array based string kernel implementations in the `kernlab` (Karatzoglou et al. (2007)) R (R Development Core Team (2008)) package and we will benchmark these string kernels. We will also briefly present the `tm` (Feinerer (2007)) R package which provides text mining functionality for R.

2 String kernels using suffix arrays

String kernels work by calculating a weighted sum over the common substring between two strings, as shown in Equation 1. Different types of kernels arise by the use of different weighting schemas. The generic form of string kernels between two sets of characters x and x' is given by the equation

$$k(x,x') = \sum_{s \sqsubseteq x, s' \sqsubseteq x'} \lambda_s \delta_{s,s'} = \sum_{s \in A^*} \mathrm{num}_s(x)\mathrm{num}_s(x')\lambda_s \ , \tag{1}$$

where A^* represents the set of all non empty strings and λ_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 whitespace character gives the "bag of words" kernel. In this paper we consider four different types of string kernels:

- Constant (constant): All common substrings are matched and weighted equally.
- Exponential decay (exponential): All common substrings are matched but the substring weight decays as the matching substring gets shorter.
- k-spectrum (spectrum): This kernel considers only matching substrings of exactly length n, i.e. $\lambda_s = 1$ for all $|s| = n$.
- Bounded range (boundrange) kernel where $\lambda_s = 0$ for all $|s| > n$ that is comparing all substrings of length less or equal to a given length n.

String kernels can be computed by building the suffix tree of a string x and computing the matching statistics of a string x' by traversing string x' through the suffix tree of x. Given a suffix tree $S(x)$ it can be proven that the occurrence of a certain substring y can be calculated by the number of nodes at the end of the path of y in the suffix tree. Auxiliary suffix links, linking identical suffixes in the tree are utilized to speed up the computations. Two main suffix tree operations are required to compute string kernels, a top down traversal for annotation and a suffix link traversal for computing matching statistics, both operations can be performed more efficiently on a suffix array.

The enhanced suffix array (Abouelhoda et al. (2004)) of a string x, is an array of integers corresponding to the lexicographically sorted suffixes of x

with additional information stored to allow for the reproduction of almost all operations available on a suffix tree. Suffix arrays bring the advantage of better memory use and locality thus most operations can be performed faster than on the original suffix trees.

3 R infrastructure

R provides a unique environment for text mining but has until recently lacked tools that would provide the necessary infrastructure in order to handle text and compute basic text related operations, e.g. the computation of a term matrix. Package `tm` provides this functionality.

3.1 `tm`

The `tm` package provides a framework for text mining applications in R. It offers functionality for managing text documents, abstracts the process of document manipulation and eases the usage of heterogeneous text formats in R. The package has integrated database backend support to minimize memory demands. An advanced metadata management is implemented for collections of text documents to alleviate the usage of large and with metadata enriched document sets. Its data structures and algorithms can be extended to fit custom demands, since the package is designed in a modular way to enable easy integration of new file formats, readers, transformations and filter operations. `tm` provides easy access to preprocessing and manipulation mechanisms such as whitespace removal, stemming, or conversion between file formats. Further a generic filter architecture is available in order to filter documents for certain criteria, or perform full text search.

3.2 `kernlab`

`kernlab` is an extensible package for kernel-based machine learning methods in R. The package contains implementations of most popular kernels, and also includes 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 (PCA).

We shortly describe two of the lesser known kernel methods we are using for our experiments.

Kernel k-means The k-means clustering algorithm has the drawback that it cannot separate clusters that are not linearly separable in input space. 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. Denoting clusters by π_j and a partitioning of points as ${\pi_j}_{j=1}^{k}$ and if Φ is the mapping function then the k-means objective function using Euclidean distances becomes

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

where $m_j = \frac{1}{\|\pi_j\|} \sum_{a \in \pi_j} \Phi(a)$ and 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 $k(a, b)$.

Spectral clustering Spectral clustering (Ng et al. (2001), Shi and Malik (2000)) works by embedding the data points of the partitioning problem into the subspace of the k largest eigenvectors of a normalized kernel matrix. The data is then embedded into the subspace of the largest eigenvectors of the normalized kernel matrix. This embedding leads to more straightforward clustering problems since points tend to form tight clusters in the eigenvector subspace.

4 Experiments

4.1 Data

Our first dataset is a subset of the Reuters-21578 dataset (Lewis (1997)) containing stories collected by the Reuters news agency. The dataset is publicly available and has been widely used in text mining research within the last decade. Our subset contains 800 documents in the category "acq" (articles on acquisitions and mergers) and 400 in the category "crude" (stories in the context of crude oil). In case of clustering this is the full dataset, for classification this is the training set. The classification test set are further 200 acquisition and 100 crude oil stories.

The second dataset is a subset of the SpamAssassin public mail corpus (`http://spamassassin.apache.org/publiccorpus/`). It is freely available and offers authentic e-mail communication with classifications into normal (ham) and unsolicited (spam) mail. For the experiment we use 800 ham documents and 400 spam documents for clustering and as training set for our classification tasks. For the latter the test set consists of additional 100 ham and 100 spam documents.

4.2 Experiment setup

The experiment setup is twofold. First, we performed clustering methods on the two datasets Reuters and SpamAssassin. We use kernel k-means and spectral clustering algorithms implemented in the `kernlab` package. We manually set the number of desired clusters to two (i.e., $k = 2$) for both datasets since in both cases we know the true labels for each document (for the Reuters set the categories acquisitions and crude oil, for the SpamAssassin set ham and spam, respectively). Secondly, we perform a "C-SVC" classification on both datasets. Again we use the SVMs in package `kernlab`.

We used four types of recent string kernels: exponential, constant, spectrum, and boundrange. We conducted runs with various values of the string length parameter used by the spectrum and bounded range kernel. We evaluate the quality of the computed cluster and classification results via the cross-agreements between the cluster or classification labels and the true labels of each dataset. In addition we measured the consumed computation time for each step. In case of clustering these are the timings for creation and computation of the string kernel matrix (denoted as Kernel Time) and the actual clustering process (denoted as Cluster Time). For classification we logged the time for string kernel matrix creation (Kernel Time), the amount of time necessary for training the SVM (SVM Training Time), and the spent time for predicting the test documents (Predict Time).

4.3 Results

Type	Length	Agree	Cluster Time	Agree	Cluster Time	Kernel Time
Algorithm		k-means		Spectral		
exponential		0.6863	1.323	0.6891	3.507	558.400
constant		0.5141	0.426	0.7958	3.604	514.080
spectrum	4	0.7766	1.346	0.8408	3.563	522.073
spectrum	6	0.8641	1.597	0.8575	3.374	519.446
spectrum	8	**0.8641**	2.365	0.8566	3.618	521.971
spectrum	10	0.6833	2.807	**0.8683**	3.669	519.317
boundrang	4	0.6900	1.472	0.6208	3.643	520.817
boundrang	6	0.6491	1.852	0.6975	3.665	520.981
boundrang	8	0.6991	1.495	0.7700	3.532	522.097
boundrang	10	0.7283	2.903	0.7816	3.486	519.738
fullstring	8			0.8266	3.851	2195.795
string	8			0.8558	3.652	4521.459

Table 1. Cross-agreement and timing results for kernel *k*-means and spectral clustering on the Reuters data for various types of string kernels set and different values for string length parameter. The string and fullstring kernels are dynamic programming based implementations of kernels similar to the spectrum and boundrange kernel. They are slower by a factor of 4 and 8 respectively.

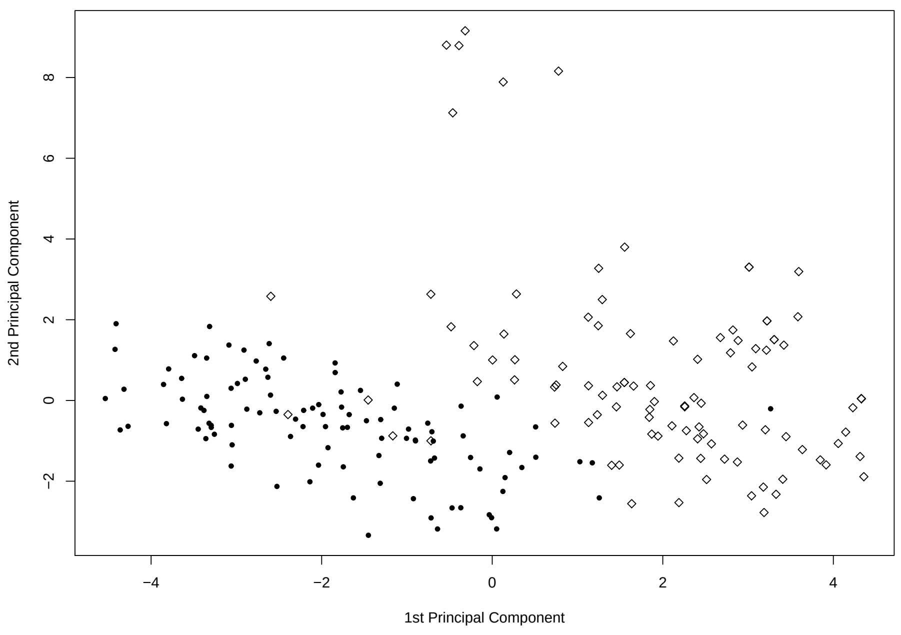

Fig. 1. The projection on two principal components of 200 Reuters text documents (100 crude in dots and 100 acquisition in diamonds) using a spectrum kernel and kernel PCA.

Type	Length	Agree	Cluster Time	Agree	Cluster Time	Kernel Time
Algorithm		k-means		Spectral		
exponential		0.7358	2.372	0.6183	3.393	1765.602
constant		0.6170	2.535	0.5991	3.635	1674.338
spectrum	4	0.6525	1.901	**0.7491**	3.351	1675.842
spectrum	6	0.6366	1.034	0.6691	3.170	1673.387
spectrum	8	0.6366	1.688	0.6691	3.147	1677.165
spectrum	10	0.5891	1.160	0.6675	3.170	1678.230
boundrang	4	**0.7750**	1.154	0.7200	3.343	1670.671
boundrang	6	0.7583	1.314	0.7300	3.353	1672.437
boundrang	8	0.7541	1.280	0.6175	3.359	1670.454
boundrang	10	0.7450	1.987	0.6158	3.378	1679.011

Table 2. Cross-agreement and timing results for kernel k-means and spectral clustering on the SpamAssassin dataset under different string kernel parameters.

Type	Length	Agree	Predict Time	Kernel Time	SVM Training Time
exponential		0.9633	0.007	562.769	0.115
constant		0.8333	0.010	516.054	0.595
spectrum	4	**0.9900**	0.005	525.716	0.102
spectrum	6	0.9800	0.006	525.044	0.128
spectrum	8	0.9766	0.007	527.880	0.141
spectrum	10	0.9566	0.008	523.306	0.168
boundrang	4	0.9600	0.007	534.470	0.118
boundrang	6	0.9666	0.006	520.825	0.117
boundrang	8	0.9633	0.008	524.199	0.108
boundrang	10	0.9633	0.007	519.974	0.114

Table 3. Cross-agreement and timing results for SVM classification on the Reuters dataset under different string kernel parameters.

Type	Length	Agree	Predict Time	Kernel Time	SVM Training Time
exponential		0.9400	0.005	1740.865	0.097
constant		0.8850	0.006	1644.471	0.199
spectrum	4	**0.9700**	0.004	1667.624	0.112
spectrum	6	0.9500	0.005	1667.321	0.127

Table 4. Cross-agreement and timing results for SVM classification on the SpamAssassin dataset under different string kernel parameters.

From the results of our experiments on the Reuters dataset (see Table 1) we can conclude that the spectrum kernel could be a generally good choice for processing text documents. It performs well overall compared to the alternative kernels. This can be attributed to the fact that it considers matches of exact length k while the other kernels match also all substrings of length smaller than k thus introducing additional information into the kernel matrix that might not be of benefit in the case of rather lengthy text documents where the probability of matches will be high and thus additional noise is introduced.

The results of the experiments on the SpamAssassin dataset (see Table 2) show that it is important to also consider the nature of the text that is being processed. In the case of the SpamAssassin dataset alternative kernels types seem to perform on the same level if not better than the spectrum kernel. This could be attributed to the particular composition of spam messages where short words occur more frequently and arbitrary text and string sequences are added into the messages, to bypass filters. Kernels such as the boundrange kernel will capture more of this information. The spectral clustering algorithm performs in a slightly more consistent way across different kernel and parameter configurations than the kernel k-means although the later seems to outperform the former for some configurations.

The results in Table 3 and Table 4 confirm the excellent performance of SVMs in text classification when using string kernels. Note that until recently string kernels have been computationally expensive and this prohibited applications in many real world datasets. This constraint has been lifted as the comparison of the performance of the suffix based kernels with a typical dynamic programming implementation shows. The new version of the kernels outperforms the older on the Reuters dataset by a factor of 4 to 8 depending on the kernel type.

Figure 1 illustrates the projection of 200 text documents from the Reuters dataset on two principal components using kernel PCA. This type of visualization can be used as an explorative tool in order to e.g. test for the existence of clusters in a set of documents.

4.4 Conclusion

We presented a fast implementation of string kernels with excellent performance in clustering and classification using spectral clustering and support vector machines. The availability of the kernels in `kernlab` along with the text processing infrastructure in the `tm` package provides R with advanced algorithms for the processing of text documents for the practitioner. In our experiments we demonstrated the excellent performance of the kernel both in term of computational cost and results and showed that the spectrum kernel is a good choice when working with normal text data while this might change when using non-text related string data like e.g. DNA sequences.

References

ABOUELHODA, M.I., KURTZ, S., OHLEBUSCH, E. (2004): Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2, 53–86.

FEINERER, I. (2007): tm: Text Mining Package: *CRAN tm version 0.3*.

HERBRICH, R. (2002): *Learning Kernel Classifiers Theory and Algorithms* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA.

KARATZOGLOU, A., SMOLA, A. and HORNIK, K. (2007): kernlab: Kernel Methods Lab *CRAN kernlab version 0.9-5*.

LEWIS, D. (1997): Reuters-21578 text categorization test collection.

NG, A., JORDAN, M. and WEISS, Y. (2001): On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems, 14*.

R Development Core Team (2008): *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

SHI, J. and MALIK, J. (2000): Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888–905.*

TEO, C.H. and VISHWANATHAN, S.V.N. (2006): Fast and space efficient string kernels using suffix arrays. *Proceedings of the 23rd International Conference on Machine learning (ICML)* ACM Press, Pittsburgh, Pennsylvania, 929–936.

VISHWANATHAN, S.V.N. and SMOLA, A.J. (2004): Fast Kernels for String and Tree Matching. In B. Schölkopf and K. Tsuda and J. P. Vert (Eds.): *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 113–130.

WATKINS, C. (2000): Dynamic Alignment Kernels. *Advances in Large Margin Classifiers*. A.J. Smola and P. L. Bartlett and B. Schölkopfand D. Schuurmans. MIT Press, Cambridge, MA, 39–50.