

Fast Text Mining Using Kernels in \mathbb{R}

Alexandros Karatzoglou¹, joint work with Ingo Feinerer²

¹INSA de Rouen, LITIS, France

²Theory and Logic Group, Institute of Computer Languages
Vienna University of Technology, Austria

COMPSTAT, 2008

Outline

- 1 Kernel Methods
- 2 String Kernel and Suffix Trees
- 3 Experiments
- 4 Results
- 5 Conclusion

Kernel-based Machine Learning

- Kernel based methods work in a feature space that is implicitly defined by a kernel.
- Usually are based on linear methods that can be transformed into non-linear methods by the definition of a proper kernel.
- Kernels can be defined directly on discrete data such as text and string sequences.
- One can then use any kernel method on the discrete data without the need for feature extraction

String Kernels

- String Kernel function

$$k(x, x') = \sum_{s \sqsubseteq x, s' \sqsubseteq x'} \lambda_s \cdot \delta_{s, s'} = \sum_{s \in A^*} num_s(x) \cdot num_s(x') \cdot \lambda_s$$

- Number of occurrences of substrings of the same length n ($\lambda_s = 0$ for all $|s| \neq n$ “spectrum”)
- Number of occurrences of substrings of length up to n ($\lambda_s = 0$ for all $|s| > n$ “Bounded range”)

String Kernel Types

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) A kernel where $\lambda_s = 0$ for all $|s| > n$ that is comparing all substrings of length less or equal to a given length n .

Efficient Computation

- 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.

Suffix Tree

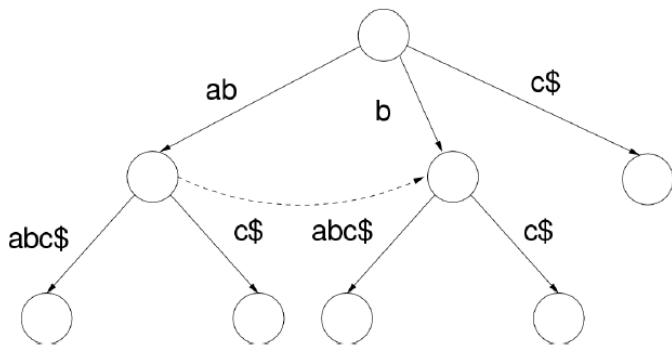


Figure: Suffix tree of the string ababc\$

Suffix Tree String kernels

- 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.
- In practise the construction of a suffix tree requires $20n$ memory where n the length of the string
- Computations scale linearly with the size of the documents $O(mn)$. But memory locality of the suffix tree makes computations rather slow and restricts the size of documents that can be used.

Suffix Arrays

- 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.

kernlab R Package

- Package for kernel-based Machine Learning methods in R
- Contains
 - 9 different kernels, Gaussian, linear, suffix array based string kernels etc.
 - kernel methods for regression and classification, novelty detection (SVM, Gaussian Processes)
 - clustering (kernel k -means, spectral clustering)
 - ranking and kernel PCA, nonparametric two sample test

tm R Package

- Text mining framework in R
- Tailored for
 - Plain texts, articles and papers
 - Web documents (XML, SGML, ...)
 - Surveys
- Methods for
 - Clustering
 - Classification
 - Visualization

Data

- We run several clustering and classification experiments using data from the Reuters-21578 dataset (Lewis (1997)) in particular 1500 news stories from the topics “acq” and “crude”. We also use a subset of the SpamAssassin public mail corpus where we select 1000 “normal” emails and 500 “spam” mails.
- We perform clustering with the Spectral and the kernel k-means algorithm in `kernlab` and classification with the Support Vector Machine in `kernlab`.

Experiment Setup

We used a wide variety of string kernels

- constant kernel
- spectrum kernel
- exponential kernel
- bounded range kernel

and parameter settings (e.g., lengths of 2, 4, 6, and 8)

Clustering Results for Reuters-21578

Type	Length	Agree	Cluster Time	Agree	Cluster Time
Algorithm		k-means		Spectral	
exponential		0.6863	1.323	0.6891	3.507
constant		0.5141	0.426	0.7958	3.604
spectrum	4	0.7766	1.346	0.8408	3.563
spectrum	6	0.8641	1.597	0.8575	3.374
spectrum	8	0.8641	2.365	0.8566	3.618
spectrum	10	0.6833	2.807	0.8683	3.669
boundrang	4	0.6900	1.472	0.6208	3.643
boundrang	6	0.6491	1.852	0.6975	3.665
boundrang	8	0.6991	1.495	0.7700	3.532
boundrang	10	0.7283	2.903	0.7816	3.486

Results for SpamAssassin

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

SVM Classification Results for Reuters-21578

Type	Length	Accuracy
exponential		0.9633
constant		0.8333
spectrum	4	0.9900
spectrum	6	0.9800
spectrum	8	0.9766
spectrum	10	0.9566
boundrang	4	0.9600
boundrang	6	0.9666
boundrang	8	0.9633
boundrang	10	0.9633

Principal Component Analysis

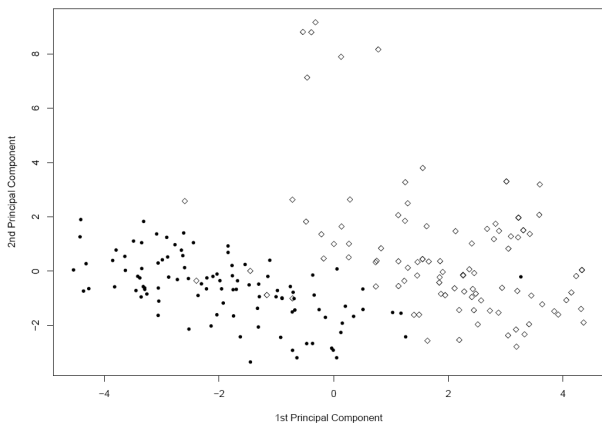


Figure: The projection on two principal components of 200 Reuters text documents (100 crude in dots and 100 acquisition in diamonds) using a spectrum kernel and kernel PCA.

Running Time

Data	Suffix Array	Simple Impl.
Spam	1670 sec.	2h
Reuters	530 sec	4200 sec.

Conclusion

- Fast implementation allows for the use of kernel methods in large text collections.
- Performance of string kernels is typically better than using term frequencies
- Implemented in `kernlab`, so you can use it :-)