# KERNELS AND SIMILARITY MEASURES FOR TEXT CLASSIFICATION

*André T. Martins*

Priberam Informática,
and
Instituto de Telecomunicações
Instituto Superior Técnico,
Lisboa, Portugal

*Mário A. T. Figueiredo*

Instituto de Telecomunicações
Instituto Superior Técnico,
Lisboa, Portugal

*Pedro M. Q. Aguiar*

Instituto de Sistemas e Robótica
Instituto Superior Técnico,
Lisboa, Portugal

## ABSTRACT

**Measuring similarity between two strings is a fundamental step in text classification and other problems of information retrieval. Recently, kernel-based methods have been proposed for this task; since kernels are inner products in a feature space, they naturally induce similarity measures. Information theoretic (dis)similarities have also been the subject of recent research. This paper describes some string kernels and information theoretic mesures and shows how they can be efficiently implemented via suffix trees. The performance of these measures is then evaluated on a text classification (authorship attribution) problem, involving a set of books by Portuguese writers.**

## 1. INTRODUCTION

Many applications in areas such as bioinformatics and natural language processing (NLP) require some kind of similarity measure between strings. In NLP, strings are sequences of alphabet characters and represent text in natural language. Text classification (*e.g.*, categorization, authorship attribution, plagiarism detection) is a class of NLP problems that require the computation of string similarities [7].

In recent years, with the emergence of kernel-based methods for pattern classification [12, 13], many string kernels have been proposed, tailored to the specificities of particular tasks and domains. Since kernels are inner products in a feature space, they naturally induce similarity measures.

In a different perspective, there has been recent interest in using information theoretic measures of (dis)similarity between sequences of symbols. Different methods distinguish themselves on how they estimate the "divergence" between the two sequences. Particularly, variations of the well-known Lempel-Ziv algorithm for compression [16, 17] have been used for this task. The idea of using compression algorithms to estimate string similarity is also present in the formal definition of the non-computable "information distance" [9] based on the algorithmic notion of Kolmogorov complexity.

The goals of this paper are the following: to provide a brief overview of string kernels and information theoretic string dissimilarities; to show how they can be implemented efficiently (in linear time) by resorting to suffix trees; finally, to assess their performance on a (Portuguese) author attribution problem. Our experiments show that all the methods considered achieve a similar and very high accuracy; the key difference is that the kernel-based measures require fine-tuning of parameters, whereas the information theoretic techniques are parameter free, thus being promising new tools for NLP problems.

## 2. STRING KERNELS

Given an input space $\mathcal{X}$, a (positive definite) *kernel* is a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying $\kappa(x, y) = \kappa(y, x)$ and

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \kappa(x_i, x_j) \geq 0, \tag{1}$$

for any $n \in \mathbb{N}$, $\{c_i\}_{i=1}^n \in \mathbb{R}^n$, and $\{x_i\}_{i=1}^n \in \mathcal{X}^n$. Given a set of points $\{x_i\}_{i=1}^n$, the $n \times n$ matrix $K = [k_{ij}] := [\kappa(x_i, x_j)]$ is called *Gram matrix*. A direct consequence of Mercer's theorem is that $\kappa$ is a (positive definite) kernel if and only if there is a *feature space* $\mathcal{F}$ endowed with an inner product $\langle ., . \rangle$ and a map $\phi : \mathcal{X} \to \mathcal{F}$ satisfying, for all $x, y \in \mathcal{X}$,

$$\kappa(x, y) = \langle \phi(x), \phi(y) \rangle. \tag{2}$$

The now famous "kernel trick" allows performing non-linear computations in the input space $\mathcal{X}$ by using linear algorithms in a feature space $\mathcal{F}$, often with high (possibly infinite) dimension, without having to explicitly compute in that feature space [12, 13]. Support vector machines (SVMs) are the best known example of the application of the kernel trick, although there are many other *kernel methods* [12, 13].

Several string kernels (*i.e.*, operating on the space of strings) were recently proposed [8, 15, 13]. Denote by $\Sigma$ the underlying alphabet, and by $\Sigma^*$ the set of all finite strings formed by characters in $\Sigma$ together with the empty string $\epsilon$. The $p$*-spectrum kernel* (PSK) [8] is associated with a feature space indexed by $\Sigma^p$ (the set of length-$p$ strings). The feature representation of a string $s$, $\Phi^p(s) \equiv (\phi_u^p(s))_{u\in\Sigma^p}$, counts the number of times each $u \in \Sigma^p$ occurs as a substring of $s$,

$$\phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|. \tag{3}$$

The PSK is then defined as the standard inner product in $\mathbb{R}^{|\Sigma|^p}$, $\kappa_p(s,t) = \langle \Phi^p(s), \Phi^p(t)\rangle$. A more general kernel is the *weighted all-substrings kernel* (WASK) [15], which takes into account the contribution of all the substrings weighted by their length. The WASK can be viewed as a convex combination of PSKs and can be written as

$$\kappa(s,t) = \sum_{p=1}^{\infty} \alpha_p \kappa_p(s,t), \tag{4}$$

where $\alpha_p$ is often chosen to decay exponentially with $p$ and truncated; for example, $\alpha_p = \lambda^p$, if $p_{\min} \le p \le p_{\max}$, and $\alpha_p = 0$, otherwise, where $0 < \lambda < 1$ is the decaying factor.

A remarkable fact (see Section 4) is that both the PSK and the WASK may be computed in $O(|s| + |t|)$ time (*i.e.*, with cost that is linear in the length of the strings) by using suffix trees. Moreover, with $s$ fixed, any kernel $\kappa(s,t)$ may be computed in time $O(|t|)$, which is particularly useful for classification applications.

## 3. INFORMATION THEORETIC DISSIMILARITIES

### 3.1. Introduction

Some compression methods, namely the Ziv-Lempel (LZ) algorithm [16, 17], are said "universal", *i.e.*, are asymptotically distribution independent. Inspired by this fact, information theoretic "universal" dissimilarities, based on the algorithms underlying "universal" compression, have been proposed [18] and used in classification problems [4], [11].

The concepts of compression and string similarity also come together in the definition of *information distance* (ID) [9]. The ID is built on the concept of *Kolmogorov complexity* or *algorithmic entropy* of a string. The idea of using compression algorithms to estimate string similarity is also present in the *GenCompress*-based algorithm to measure the relatedness of two DNA sequences [2], and in a recent approach that uses the Burrows-Wheeler transform [1].

### 3.2. Kullback-Leibler divergence

Let $p$ and $q$ be two probability functions of discrete random processes (discrete sources, either memoryless stationary or Markovian of arbitrary order) with values in an alphabet $\Sigma$. Let the *Kullback-Leibler divergence* (DKL) between $p$ and $q$ be denoted as $D_{KL}(p\|q)$ (see [3] for formal expressions and further details). The DKL may be regarded as a dissimilarity measure between $p$ and $q$, since $D_{KL}(p\|q) \ge 0$ and $D_{KL}(p\|q) = 0$ if and only if $p = q$. However, it is not a metric, due to lack of symmetry and failure to satisfy the triangle inequality. The *Jensen-Shannon divergence* (JSD), defined as

$$D_{JS}(p\|q) = D_{KL}\left(p\middle\|\frac{p+q}{2}\right) + D_{KL}\left(q\middle\|\frac{p+q}{2}\right), \tag{5}$$

was recently proved to be the square of a metric [5].

### 3.3. Lempel-Ziv parsing and entropy estimation

Consider a string $x \in \Sigma^n$ emitted by a stationary source. Let $c(x)$ denote the number of phrases in $x$ resulting from the sequential LZ parsing of $x$ into distinct phrases, *i.e.*, such that each phrase is the shortest string which is not a previously parsed phrase. When $n \to \infty$, the average LZ code length for $x$ may be approximated by

$$\frac{1}{n} c(x) \log_2 c(x); \tag{6}$$

it is well-known that this converges almost surely (a.s.) to the entropy rate of the source producing $x$ [3]. This suggests using the output of a LZ encoder as an estimate of the entropy of a stationary source, without estimating any model parameters.

### 3.4. The Ziv-Merhav method

The idea of using a LZ encoder as an entropy estimator was extended by Ziv and Merhav (ZM) [18] to estimate relative entropy, by using a variation of the LZ algorithm to perform the "cross-parsing" of two strings. Let $z$ and $x$ be two strings of length $n$. First, $z$ is parsed by the incremental LZ parsing algorithm into $c(z)$ distinct phrases (see the previous section); *e.g.*, if $n = 11$ and $z = abbbbaaabba$, then the self incremental parsing yields $a|b|bb|ba|aa|bba$, that is, $c(z) = 6$. Then, one applies a variation of the LZ algorithm which performs a sequential cross-parsing of $z$ with respect to $x$: each phrase is the longest substring of $z$ which was parsed in $x$. For example, if $x = baababaabba$, parsing $z$ with respect to $x$ yields $abb|bba|aabba$, that is, $c(z|x) = 3$. It was proved in [18] that, for two sequences $x, z$ of length $n \to \infty$, produced by two Markovian sources, the quantity

$$\Delta(z\|x) = \frac{1}{n}[c(z|x)\log_2 n - c(z)\log_2 c(z)] \tag{7}$$

converges a.s. to the DKL between the sources. Notice that $[c(z)\log_2 c(z)]/n$ can be seen as a measure of the entropy of the source that emitted $z$, while $[c(z|x)\log_2 n]/n$ provides an estimate of the code length obtained when coding $z$ using a model for $x$. In Section 4 we show how the ZM method can be implemented in linear time (i.e., $O(|x| + |z|)$) using suffix trees.

## 4. IMPLEMENTATION ASPECTS

### 4.1. Suffix trees

Basically, a *suffix tree* (ST) it is a data structure containing all the suffixes of a given string $s$, that allows answering queries such as “is $t$ a substring of $s$?” in time $O(|t|)$. The ST for $s$ may be built in time $O(|s|)$ using, *e.g.*, Ukkonen’s algorithm [14]. A *generalized ST* (GST) for strings $s_1, \ldots, s_n$ contains all the suffixes of these strings and may be built in time $O(|s_1| + \ldots + |s_n|)$. Reference [6] is a comprehensive text on STs and related data structures and algorithms.

### 4.2. Implementing the PSK and WASK with suffix trees

Let $s$ and $t$ be strings whose similarity one wants to measure; a GST for $s\$$ and $t\#$ (where \$ and $\#$ are unique terminating characters) allows computing the PSK and the WASK in time $O(|s| + |t|)$. We first count, for each node $v$ in the tree, how many times the string path $S(v)$ occurs as a substring in $s$ and $t$. Denoting these occurrences as $n_s(v)$ and $n_t(v)$, respectively, this can be done recursively, *i.e.*,

$$n_s(v) = \begin{cases} 1 & \text{if } v \text{ is a \$-leaf,} \\ 0 & \text{if } v \text{ is a \#-leaf,} \\ \sum_{w \in \mathrm{Child}(v)} n_s(w) & \text{otherwise,} \end{cases} \tag{8}$$

and analogously for $n_t(v)$. Now, let $\pi(v)$ denote the parent node of $v$ if $v$ is not the root. There is a $p$-gram in the edge that connects $\pi(v)$ to $v$ if and only if $|S(\pi(v))| < p \leq |S(v)|$. Define the set $\mathcal{V}_p = \{v : |S(\pi(v))| < p \leq |S(v)|\}$. The PSK is obtained with overall complexity $O(|s| + |t|)$ via

$$\kappa_p(s,t) = \sum_{v \in \mathcal{V}_p} n_s(v)\, n_t(v). \tag{9}$$

If we use the weights $\alpha_p$ defined in Section 2, also the WASK may be computed in time $O(|s| + |t|)$ through

$$\kappa(s,t) = \sum_{v \in \mathcal{V}} n_s(v) n_t(v) \beta(v), \tag{10}$$

where $\mathcal{V}$ is the set of all nodes, $\beta(v) = \sum_{p=p_0(v)}^{p_f(v)} \lambda^p$, $p_0(v) = \max\{p_{\min}, |S(\pi(v))| + 1\}$, and $p_f(v) = \min\{p_{\max}, |S(v)|\}$. Note that $\beta(v)$ is easily computed in $O(1)$.

### 4.3. Implementing the ZM method with suffix trees

The two building blocks of the ZM method (LZ parsing and LZ-type cross parsing) can both be implemented using STs. LZ parsing of a string $z$, based on a ST is described in [6]: at stage $j$, we want to obtain the longest parsed phrase (say, with length $l_j$) plus the subsequent character. By building a ST for $z$, and preprocessing it by writing at each node the lowest position in $z$ that achieves it, the sequence of stages $1, \ldots, c(z)$ may be performed by successively querying the ST in time $O(l_j)$. Hence the overall complexity is $O(\sum_{j=1}^{c(z)} l_j) = O(|z|)$. Cross parsing can be done analogously using only a ST for $x$, the only difference being that we query it with substrings of $z$ (this corresponds to a partial computation of the matching statistics of $z$ with respect to $x$). The overall complexity is also $O(|x| + |z|)$; however, in practice, ZM is computationally cheaper than PSK and WASK, since it doesn’t require a GST.

## 5. EXPERIMENTS

We have evaluated the performance of the above described methods on a text authorship attribution task, using a set of texts from Portuguese writers (available at the Project Gutenberg site `www.gutenberg.org`). Since for most authors there is only one book available, each book was split into passages of about $50$ KB each. The Gram matrices and the (symmetrized) matrix of DKL estimates were then computed using the described ST-based methods. Each entry in these matrices is indexed by a pair of passages. Distance matrices are computed from the Gram matrices via the cosine measure. For the WASK, we let $p_{\min} = p$ vary and fix $p_{\max} = \infty$ (no upper bound on the length of the string features, as in [15]).

The nearest neighbor rule was used: each passage is attributed to the author of the closest passage excluding itself. Table 1 shows the best results achieved by each method.

**Table 1**. Results of the text authorship attribution task using the PSK, WASK and ZM methods.

| **Author** | **No. pass.** | **No. correct attributions** | | |
|---|---|---|---|---|
| | | PSK | WASK | ZM |
| | | $p = 5$ | $p = 4, \lambda = 0.5$ | |
| A. Herculano | 37 | 37 | 37 | 36 |
| A. Nobre | 2 | 2 | 2 | 2 |
| C. C. Branco | 5 | 5 | 5 | 5 |
| E. Queiroz | 2 | 2 | 2 | 2 |
| F. Lopes | 4 | 4 | 4 | 4 |
| G. Junqueiro | 3 | 3 | 3 | 3 |
| J. Dinis | 30 | 30 | 30 | 30 |
| J. Mattos | 5 | 5 | 5 | 5 |
| L. Netto | 2 | 2 | 2 | 2 |
| L. V. Camões | 6 | 6 | 6 | 6 |
| P. Silva | 6 | 4 | 4 | 5 |
| N. Tolentino | 2 | 2 | 2 | 2 |
| R. Pina | 4 | 4 | 4 | 4 |
| R. Ortigão | 17 | 17 | 17 | 17 |
| **Tot.** | **125** | **123** | **123** | **123** |
| **Acc. (%)** | | **98.4** | **98.4** | **98.4** |

The ZM method, which does not require tuning any parameters, yields the same accuracy (98.4%) as the best fine-tuned ($p$ and $\lambda$) kernel-based methods. For the PSK, a too large value of $p$ (*e.g.*, $p = 10$) strongly decreases the performance of the algorithm, which is understandable: if $p$ is larger than the length of the greatest common substrings, the kernel becomes zero. On the other side, small values of $p$ discard important features, such as long substrings. The ZM “distance”, in contrast, has an “unbounded” capacity to deal with long substrings, since there are no bounds on the length of the phrases obtained during the cross-parsing.

Interestingly, one of the (only two) passages misclassified by the ZM classifier revealed a strong connection between two texts in the collection. In fact, Herculano’s book *Opúsculos* is analyzed in an edition of *As Farpas*, by Ortigão, which can explain why it's misclassified as having been written by Ortigão. This fact suggests the ability of the ZM approach to handle citation or plagiarism detection.

We used the whole books of each author to build a phylogenetic tree of Portuguese authors, using the ZM method. The idea is to characterize the "style" of each author based on information theoretic methods. The phylogenetic tree in Figure 1 was obtained using the phylogeny inference package PHYLIP (see `http://evolution.genetics.washington.edu/phylip.html`), which basically constructs a tree by minimizing the net disagreement between the matrix pairwise distances and the distances measured on the tree. Notice the ability of this method to discriminate among 15th/16th century chroniclers (Fernão Lopes, Ruy de Pina), 19th century novelists (Camilo Castelo Branco, Júlio Dinis, Guerra Junqueiro, Eça de Queiroz, Ramalho Ortigão), 19th century essayists or historians (Alexandre Herculano, Possidónio da Silva, Júlio de Mattos) and poets (Luís de Camões, Nicolau Tolentino, Cesário Verde, António Nobre), with the sole exception of Lopes Netto, a 20th century Brazilian novelist who is misplaced too close to ancient poets perhaps due to some similarity between Brazilian Portuguese and earlier European Portuguese.

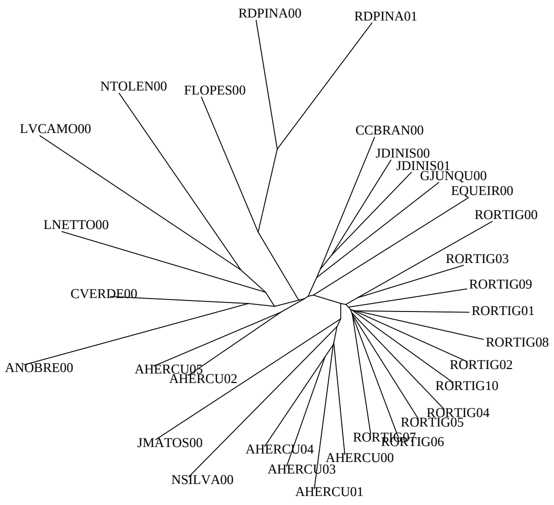

**Fig. 1**. Phylogenetic tree of Portuguese authors and books obtained by Ziv-Merhav method.

## 6. CONCLUSIONS

After reviewing some kernel-based and information theoretic methods for measuring string (dis)similarity (the $p$-spectrum kernel, the weighted all substrings kernel, and the Ziv-Merhav method) we have shown how these methods can be efficiently implemented using suffix trees. The methods were then tested on a text authorship attribution problem, all yielding very high accuracies, provided the kernel parameters are properly fine-tuned. The ZM method does not require any tuning, which is an important advantage.

In future work we will further study the relationship between (relative) entropy and (generalized) suffix trees, verifying if some information theoretic measures are "kernelizable". We will also consider other dissimilarity measures, such as the Jensen-Shannon divergence.

## 7. REFERENCES

[1] H. Cai, S. Kulkarni, S. Verdú. "Universal entropy estimation via block sorting." *IEEE Trans. Inform. Theory*, vol. 50, pp. 1551–1561, 2004.

[2] X. Chen, S. Kwong, M. Li. "A compression algorithm for DNA sequences and its applications in genome comparison." *Res. in Comput. Molec. Biol. – RECOMB'2000*, pp. 107, 2000.

[3] T. Cover, J. Thomas. *Elements of Inform. Theory*. Wiley, 1991.

[4] R. El-Yaniv, S. Fine, N. Tishby. "Agnostic classification of markovian sequences." *Neural Information Processing Systems*, MIT Press, 1997.

[5] D. Endres, J. Schindelin. "A new metric for probability distributions." *IEEE Trans. Inform. Theory*, vol. 49, pp. 1858–1860, 2003.

[6] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[7] T. Joachims. *Learning to Classify Text using Support Vector Machines*, Kluwer, 2002.

[8] C. Leslie, E. Eskin, W. Stafford-Noble. "The spectrum kernel: A string kernel for SVM protein classification." *Proc. Pacific Symp. on Biocomputing 2002*, pp. 564–575, 2002.

[9] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi. "The similarity metric. " *IEEE Trans. Inform. Theo.*, vol. 50, pp. 3250–3264, 2004.

[10] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins. "Text classification using string kernels." *Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.

[11] D. Pereira-Coutinho, M. Figueiredo. "Information theoretic text classification using the Ziv-Merhav method." In *Proc. Iberian Conf. Patt. Rec. and Image Anal.*, pp. 355–362, 2005.

[12] B. Schölkopf, A. Smola. *Learning with Kernels*. The MIT Press, 2002.

[13] J. Shawe-Taylor, N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[14] E. Ukkonen. "On-line construction of suffix trees." *Algorithmica*, vol. 14, pp. 249–260, 1995.

[15] S. Vishwanathan, A. Smola. "Fast kernels for string and tree matching." In K. Tsuda, B. Schölkopf, and J.P. Vert, editors, *Kernels and Bioinformatics*, MIT Press, 2003.

[16] J. Ziv, A. Lempel. "A universal algorithm for sequential data compression." *IEEE Trans. Inform. Theory*, vol. 23, pp. 337–343, 1977.

[17] J. Ziv, A. Lempel. "Compression of individual sequences via variable-rate coding." *IEEE Trans. Inform. Theory*, vol. 24, pp. 530–536, 1978.

[18] J. Ziv, N. Merhav. "A measure of relative entropy between individual sequences with application to universal classification." *IEEE Trans. Inform. Theory*, 39, pp. 1270–1279, 1993.