The Story of the Characters, the DNA and the Native Language

Marius Popescu
University of Bucharest
Department of Computer Science
Academiei 14, Bucharest, Romania
popescunmarius@gmail.com

Radu Tudor Ionescu
University of Bucharest
Department of Computer Science
Academiei 14, Bucharest, Romania
raducu.ionescu@gmail.com

Abstract

This paper presents our approach to the 2013 Native Language Identification shared task, which is based on machine learning methods that work at the character level. More precisely, we used several string kernels and a kernel based on Local Rank Distance (LRD). Actually, our best system was a kernel combination of string kernel and LRD. While string kernels have been used before in text analysis tasks, LRD is a distance measure designed to work on DNA sequences. In this work, LRD is applied with success in native language identification.

Finally, the Unibuc team ranked third in the closed NLI Shared Task. This result is more impressive if we consider that our approach is language independent and linguistic theory neutral.

1 Introduction

This paper presents our approach to the shared task on Native Language Identification, NLI 2013. We approached this task with machine learning methods that work at the character level. More precisely, we treated texts just as sequences of symbols (strings) and used different string kernels in conjunction with different kernel-based learning methods in a series of experiments to assess the best performance level that can be achieved. Our aim was to investigate if identifying native language is possible with machine learning methods that work at the character level. By disregarding features of natural language such as words, phrases, or meaning, our approach has an important advantage in that it is language independent.

Using words is natural in text analysis tasks like text categorization (by topic), authorship identification and plagiarism detection. Perhaps surprisingly, recent results have proved that methods handling the text at character level can also be very effective in text analysis tasks (Lodhi et al., 2002; Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Grozea et al., 2009; Popescu, 2011; Popescu and Grozea, 2012). In (Lodhi et al., 2002) string kernels were used for document categorization with very good results. Trying to explain why treating documents as symbol sequences and using string kernels led to such good results the authors suppose that: "the [string] kernel is performing something similar to stemming, hence providing semantic links between words that the word kernel must view as distinct". String kernels were also successfully used in authorship identification (Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Popescu and Grozea, 2012). For example, the system described in (Popescu and Grozea, 2012) ranked first in most problems and overall in the PAN 2012 Traditional Authorship Attribution tasks. A possible reason for the success of string kernels in authorship identification is given in (Popescu and Dinu, 2007): "the similarity of two strings as it is measured by string kernels reflects the similarity of the two texts as it is given by the short words (2-5 characters) which usually are function words, but also takes into account other morphemes like suffixes ('ing' for example) which also can be good indicators of the author's style".

Even more interesting is the fact that two methods, that are essentially the same, obtained very good results for text categorization (by topic) (Lodhi et al., 2002) and authorship identification (Popescu and Dinu, 2007). Both are based on SVM and a string kernel of length 5. How is this possible? Traditionally, the two tasks, text categorization (by topic) and authorship identification are viewed as opposite. When words are considered as features, for text categorization the (stemmed) content words are used (the stop words being eliminated), while for authorship identification the function words (stop words) are used as features, the others words (content words) being eliminated. Then, why did the same string kernel (of length 5) work well in both cases? In our opinion the key factor is the kernel-based learning algorithm. The string kernel implicitly embeds the texts in a high dimensional feature space, in our case the space of all (sub)strings of length 5. The kernel-based learning algorithm (SVM or another kernel method), aided by regularization, implicitly assigns a weight to each feature, thus selecting the features that are important for the discrimination task. In this way, in the case of text categorization the learning algorithm (SVM) enhances the features (substrings) representing stems of content words, while in the case of authorship identification the same learning algorithm enhances the features (substrings) representing function words.

Using string kernels will make the corresponding learning method completely language independent, because the texts will be treated as sequences of symbols (strings). Methods working at the word level or above very often restrict their feature space according to theoretical or empirical principles. For example, they select only features that reflect various types of spelling errors or only some type of words, such as function words, for example. These features prove to be very effective for specific tasks, but other, possibly good features, depending on the particular task, may exist. String kernels embed the texts in a very large feature space (all substrings of length k) and leave it to the learning algorithm (SVM or others) to select important features for the specific task, by highly weighting these features.

A method that considers words as features can not be language independent. Even a method that uses only function words as features is not completely language independent because it needs a list of function words (specific to a language) and a way to segment a text into words which is not an easy task for some languages, like Chinese.

Character n-grams were already used in native language identification (Brooke and Hirst, 2012; Tetreault et al., 2012). The reported performance when only character n-grams were used as features was modest compared with other type of features. But, in the above mentioned works, the authors investigated only the bigrams and trigrams and not longer n-grams. Particularly, we have obtained similar results with (Tetreault et al., 2012) when using character bigrams, but we have achieved the best performance using a range of 5 to 8 n-grams (see section 4.3). We have used with success a similar approach for the related task of identifying translationese (Popescu, 2011).

The first application of string kernel ideas came in the field of text categorization, with the paper (Lodhi et al., 2002), followed by applications in bioinformatics (Leslie et al., 2002). Computer science researchers have developed a wide variety of methods that can be applied with success in computational biology. Such methods range from clustering techniques used to analyze the phylogenetic trees of different organisms (Dinu and Sgarro, 2006; Dinu and Ionescu, 2012b), to genetic algorithms used to find motifs or common patterns in a set of given DNA sequences (Dinu and Ionescu, 2012a). Most of these methods are based on a distance measure for strings, such as Hamming (Chimani et al., 2011; Vezzi et al., 2012), edit (Shapira and Storer, 2003), Kendall-tau (Popov, 2007), or rank distance (Dinu, 2003). A similar idea to character n-grams was introduced in the early years of bioinformatics, where k-mers are used instead of single characters [1]. There are recent studies that use k-mers for the phylogenetic analysis of organisms (Li et al., 2004), or for sequence alignment (Melsted and Pritchard, 2011). Analyzing DNA at substring level is also more suited from a biological point of view, because DNA substrings may contain meaningful information. For example, genes are encoded by a number close to 100 base pairs, or codons that encode the twenty standard amino acids are formed of 3-mers. Local Rank Dis-

[1]In biology, single DNA characters are also referred to as nucleotides or monomers. Polymers are also known as k-mers.

tance (LRD) (Ionescu, 2013) has been recently proposed as an extension of rank distance. LRD drops the annotation step of rank distance, and uses k-mers instead of single characters. The work (Ionescu, 2013) shows that LRD is a distance function and that it has very good results in phylogenetic analysis and DNA sequence comparison. But, LRD can be applied to any kind of string sequences, not only to DNA. Thus, LRD was transformed into a kernel and used for native language identification. Despite the fact it has no linguistic motivation, LRD gives surprisingly good results for this task. Its performance level is lower than string kernel, but LRD can contribute to the improvement of string kernel when the two methods are combined.

The paper is organized as follows. In the next section, the kernel methods we used are briefly described. Section 3 presents the string kernels and the LRD, and shows how to transform LRD into a kernel. Section 4 presents details about the experiments. It gives details about choosing the learning method, parameter tuning, combining kernels and results of submitted systems. Finally, conclusions are given in section 5.

2 Kernel Methods and String Kernels

Kernel-based learning algorithms work by embedding the data into a feature space (a Hilbert space), and searching for linear relations in that space. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly.

Given an input set $\mathcal{X}$ (the space of examples), and an embedding vector space $\mathcal{F}$ (feature space), let $\phi : \mathcal{X} \rightarrow \mathcal{F}$ be an embedding map called feature map.

A *kernel* is a function k, such that for all $x, z \in \mathcal{X}$, $k(x, z) =< \phi(x), \phi(z) >$, where $< \cdot, \cdot >$ denotes the inner product in $\mathcal{F}$.

In the case of binary classification problems, kernel-based learning algorithms look for a discriminant function, a function that assigns $+1$ to examples belonging to one class and -1 to examples belonging to the other class. This function will be a linear function in the space $\mathcal{F}$, that means it will have the form:

$$f(x) = \text{sign}(< w, \phi(x) > +b),$$

for some weight vector w. The kernel can be exploited whenever the weight vector can be expressed as a linear combination of the training points, $\sum\limits_{i=1}^{n} \alpha_i \phi(x_i)$, implying that f can be expressed as follows:

$$f(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i k(x_i, x) + b).$$

Various kernel methods differ by the way in which they find the vector w (or equivalently the vector α). Support Vector Machines (SVM) try to find the vector w that defines the hyperplane that maximally separates the images in $\mathcal{F}$ of the training examples belonging to the two classes. Mathematically, SVMs choose the w and the b that satisfy the following optimization criterion:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(< w, \phi(x_i) > +b)]_+ + \nu ||w||^2$$

where y_i is the label ($+1$/-1) of the training example x_i, ν a regularization parameter and $[x]_+ = \max(x, 0)$.

Kernel Ridge Regression (KRR) selects the vector w that simultaneously has small empirical error and small norm in Reproducing Kernel Hilbert Space generated by kernel k. The resulting minimization problem is:

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} (y_i - < w, \phi(x_i) >)^2 + \lambda ||w||^2$$

where again y_i is the label ($+1$/-1) of the training example x_i, and λ a regularization parameter.

Details about SVM and KRR can be found in (Taylor and Cristianini, 2004). The important fact is that the above optimization problems are solved in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products which in turn are given by the kernel function k.

3 String Kernels and Local Rank Distance

The kernel function offers to the kernel methods the power to naturally handle input data that are not in the form of numerical vectors, for example strings. The kernel function captures the intuitive notion of similarity between objects in a specific domain and can be any function defined on the respective domain that is symmetric and positive definite. For strings, many such kernel functions exist with various applications in computational biology and computational linguistics (Taylor and Cristianini, 2004).

3.1 String Kernels

Perhaps one of the most natural ways to measure the similarity of two strings is to count how many substrings of length p the two strings have in common. This gives rise to the p-spectrum kernel. Formally, for two strings over an alphabet Σ, $s, t \in \Sigma^*$, the p-spectrum kernel is defined as:

$$k_p(s,t) = \sum_{v \in \Sigma^p} \text{num}_v(s) \cdot \text{num}_v(t)$$

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s [2]. The feature map defined by this kernel associates to each string a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p (p-grams).

A variant of this kernel can be obtained if the embedding feature map is modified to associate to each string a vector of dimension $|\Sigma|^p$ containing the presence bits (instead of frequencies) of all its substrings of length p. Thus the character p-grams presence bits kernel is obtained:

$$k_p^{0/1}(s,t) = \sum_{v \in \Sigma^p} \text{in}_v(s) \cdot \text{in}_v(t)$$

where $\text{in}_v(s)$ is 1 if string v occurs as a substring in s and 0 otherwise.

Normalized versions of these kernels ensure a fair comparison of strings of different lengths:

$$\hat{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s) \cdot k_p(t,t)}}$$

$$\hat{k}_p^{0/1}(s,t) = \frac{k_p^{0/1}(s,t)}{\sqrt{k_p^{0/1}(s,s) \cdot k_p^{0/1}(t,t)}}.$$

Taking into account p-grams of different length and summing up the corresponding kernels, new kernels (called *blended spectrum kernels*) can be obtained.

[2]Note that the notion of substring requires contiguity. See (Taylor and Cristianini, 2004) for a discussion about the ambiguity between the terms *substring* and *subsequence* across different traditions: biology, computer science.

3.2 Local Rank Distance

Local Rank Distance is an extension of rank distance that drops the annotation step and uses n-grams instead of single characters. Thus, characters in one string are simply matched with the nearest similar characters in the other string. To compute the LRD between two strings, the idea is to sum up all the offsets of similar n-grams between the two strings. For every n-gram in one string, we search for a similar n-gram in the other string. First, look for similar n-grams in the same position in both strings. If those n-grams are similar, sum up 0 since there is no offset between them. If the n-grams are not similar, start looking around the initial n-gram position in the second string to find an n-gram similar to the one in the first string. If a similar n-gram is found during this process, sum up the offset between the two n-grams. The search goes on until a similar n-gram is found or until a maximum offset is reached. LRD is formally defined next.

Definition 1 *Let $S_1, S_2 \in \Sigma^*$ be two strings with symbols (n-grams) from the alphabet Σ. Local Rank Distance between S_1 and S_2 is defined as:*

$$\begin{aligned}\Delta_{LRD}(S_1,S_2) &= \Delta_{left} + \Delta_{right}\\ &= \sum_{x_s \in S_1} \min_{x_s \in S_2} \{|pos_{S_1}(x_s) - pos_{S_2}(x_s)|, m\} +\\ &+ \sum_{y_s \in S_2} \min_{y_s \in S_1} \{|pos_{S_1}(y_s) - pos_{S_2}(y_s)|, m\},\end{aligned}$$

where x_s and y_s are occurrences of symbol $s \in \Sigma$ in strings S_1 and S_2, $pos_S(x_s)$ represents the position (or the index) of the occurrence x_s of symbol $s \in \Sigma$ in string S, and $m \geq 1$ is the maximum offset.

A string may contain multiple occurrences of a symbol $s \in \Sigma$. LRD matches each occurrence x_s of symbol $s \in \Sigma$ from a string, with the nearest occurrence of symbol s in the other string. A symbol can be defined either as a single character, or as a sequence of characters (n-grams). Overlapping n-grams are also permitted in the computation of LRD. Notice that in order to be a symmetric distance measure, LRD must consider every n-gram in both strings. The complexity of an algorithm to compute LRD can be reduced to $O(l \times m)$ using advanced string searching algorithms, where l is the maximum length of the two strings involved in the computation of LRD, and m is the maximum offset.

To understand how LRD actually works, consider example 1 where LRD is computed between strings s_1 and s_2 using 1-grams (single characters).

Example 1 *Let* $s_1 = CCBAADACB$, $s_2 = DBACDCA$, *and* $m = 10$ *be the maximum offset. The LRD between* s_1 *and* s_2 *is given by:*

$$\Delta_{LRD}(s_1, s_2) = \Delta_{left} + \Delta_{right}$$

where the two sums Δ_{left} *and* Δ_{right} *are computed as follows:*

$$\Delta_{left} = \sum_{x_s \in s_1} \min_{x_s \in s_2} \{|pos_{s_1}(x_s) - pos_{s_2}(x_s)|, 10\}$$
$$= |1-4| + |2-4| + |3-2| + |4-3| + |5-3| +$$
$$+ |6-5| + |7-7| + |8-6| + |9-2| = 19$$

$$\Delta_{right} = \sum_{y_s \in s_2} \min_{y_s \in s_1} \{|pos_{s_1}(y_s) - pos_{s_2}(y_s)|, 10\}$$
$$= |1-6| + |2-3| + |3-4| + |4-2| + |5-6| +$$
$$+ |6-8| + |7-7| = 12.$$

In other words, Δ_{left} *considers every symbol from* s_1*, while* Δ_{right} *considers every symbol from* s_2*. Observe that* $\Delta_{LRD}(s_1, s_2) = \Delta_{LRD}(s_2, s_1)$.

LRD measures the distance between two strings. Knowing the maximum offset (used to stop similar n-gram searching), the maximum LRD value between two strings can be computed as the product between the maximum offset and the number of pairs of compared n-grams. Thus, LRD can be normalized to a value in the $[0, 1]$ interval. By normalizing, LRD is transformed into a dissimilarity measure. LRD can be also used as a kernel, since kernel methods are based on similarity. The classical way to transform a distance or dissimilarity measure into a similarity measure is by using the Gaussian-like kernel (Taylor and Cristianini, 2004):

$$k(s_1, s_2) = e^{-\dfrac{LRD(s_1, s_2)}{2\sigma^2}}$$

where s_1 and s_2 are two strings. The parameter σ is usually chosen to match the number of features (characters) so that values of $k(s_1, s_2)$ are well scaled.

4 Experiments

4.1 Dataset

The dataset for the NLI shared task is the TOEFL11 corpus (Blanchard et al., 2013). This corpus contains 9900 examples for training, 1100 examples for development (or validation) and another 1100 examples for testing. Each example is an essay written in English by a person that is a non-native English speaker. The people that produced the essays have one of the following native languages: German, French, Spanish, Italian, Chinese, Korean, Japanese, Turkish, Arabic, Telugu, Hindi. For more details see (Blanchard et al., 2013).

We participated only in the closed NLI shared task, where the goal of the task is to predict the native language of testing examples, only by using the training and development data. In our approach, documents or essays from this corpus are treated as strings. Thus, when we refer to *strings* throughout this paper, we really mean *documents* or *essays*. Because we work at the character level, we didn't need to split the texts into words, or to do any NLP-specific preprocessing. The only editing done to the texts was the replacing of sequences of consecutive space characters (space, tab, new line, etc.) with a single space character. This normalization was needed in order to not artificially increase or decrease the similarity between texts as a result of different spacing. Also all uppercase letters were converted to the corresponding lowercase ones. We didn't use the additional information from *prompts* and English language proficiency level.

4.2 Choosing the Learning Method

SVM and KRR produce binary classifiers and native language identification is a multi-class classification problem. There are a lot of approaches for combining binary classifiers to solve multi-class problems. Typically, the multiclass problem is broken down into multiple binary classification problems using common decomposing schemes such as: one-versus-all (OVA) and one-versus-one (OVO). There are also kernel methods that directly take into account the multiclass nature of the problem such as the kernel partial least squares regression (KPLS).

We conducted a series of preliminary experiments in order to select the learning method. In these ex-

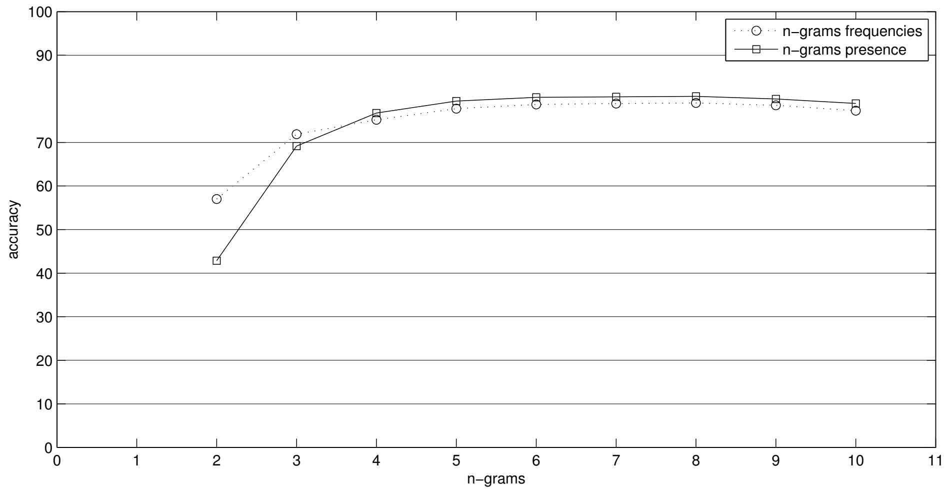

Figure 1: 10-fold cross-validation accuracy on the train set for different n-grams.

Method	Accuracy
OVO SVM	72.72%
OVA SVM	74.94%
OVO KRR	73.99%
OVA KRR	$\mathbf{77.74\%}$
KPLS	74.99%

Table 1: Accuracy rates using 10-fold cross-validation on the train set for different kernel methods with $\hat{k}_5$ kernel.

periments we fixed the kernel to the p-spectrum normalized kernel of length 5 ($\hat{k}_5$) and plugged it in the following learning methods: OVO SVM, OVA SVM, OVO KRR, OVA KRR and KPLS. Note that in this stage we were interested only in selecting the learning method and not in finding the best kernel. We chose the $\hat{k}_5$ because it was reported to work well in the case of the related task of identifying translationese (Popescu, 2011).

We carried out a 10-fold cross-validation on the training set and the result obtained (with the best parameters setting) are shown in Table 1.

The results show that for native language identification the one-vs-all scheme performs better than the one-versus-one scheme. The same fact was reported in (Brooke and Hirst, 2012). See also (Rifkin and Klautau, 2004) for arguments in favor of one-vs-all. The best result was obtained by one-vs-all Kernel Ridge Regression and we selected it as our learning method.

4.3 Parameter Tuning for String Kernel

To establish the type of kernel, (blended) p-spectrum kernel or (blended) p-grams presence bits kernel, and the length(s) of of n-grams that must be used, we performed another set of experiments. For both p-spectrum normalized kernel and p-grams presence bits normalized kernel, and for each value of p from 2 to 10, we carried out a 10-fold cross-validation on the train set. The results are summarized in Figure 1. As can be seen, both curves have similar shapes, both achieve their maximum at 8, but the accuracy of the p-grams presence bits normalized kernel is generally better than the accuracy of the p-spectrum normalized kernel. It seem that in native language identification the information provided by the presence of an n-gram is more important than the the information provided by the frequency of occurrence of the respective n-gram. This phenomenon was also noticed in the context of sexual predator identification (Popescu and Grozea, 2012).

We also experimented with different blended kernels to see if combining n-grams of different lengths can improve the accuracy. The best result was obtained when all the n-grams with the length in the range 5-8 were used, that is the 5-8-grams presence bits normalized kernel ($\hat{k}^{0/1}_{5-8}$). The 10-fold cross-validation accuracy on the train set for this kernel

Method	Accuracy
KRR + K_{LRD_6}	42.1%
KRR + K_{nLRD_4}	70.8%
KRR + K_{nLRD_6}	74.4%
KRR + K_{nLRD_8}	$\mathbf{74.8}\%$

Table 2: Accuracy rates, using 10-fold cross-validation on the training set, of LRD with different n-grams, with and without normalization. Normalized LRD is much better.

was 80.94% and was obtained for the KRR parameter λ set to 10^{-5}. The authors of (Bykh and Meurers, 2012) also obtained better results using n-grams with the length in a range than using n-grams of a fixed length.

4.4 Parameter Tuning for LRD Kernel

Parameter tuning for LRD kernel (K_{LRD}) was also done by using 10-fold cross validation on the training data. First, we observed that the KRR based on LRD works much better with the normalized version of LRD (K_{nLRD}). Another concern was to choose the right length of n-grams. We tested with several n-grams such as 4-grams, 6-grams and 8-grams that are near the mean English word length of 5-6 letters. The tests show that the LRD kernels based on 6-grams (K_{nLRD_6}) and 8-grams (K_{nLRD_8}) give the best results. In the end, the LRD kernels based on 6-grams and 8-grams are combined to obtain even better results (see section 4.5). Finally, the maximum offset parameter m involved in the computation of LRD was chosen so that it generates search window size close to the average number of letters per document from the TOEFL 11 set. There are 1802 characters per document on average, and m was chosen to be 700. This parameter was also chosen with respect to the computational time of LRD, which is proportional to the parameter value. Table 2 shows the results of the LRD kernel with different parameters cross validated on the training set. For K_{nLRD}, the σ parameter of the Gaussian-like kernel was set to 1. The reported accuracy rates were obtained with the KRR parameter λ set to 10^{-5}.

Regarding the length of strings, we observed that LRD is affected by the variation of string lengths. When comparing two documents with LRD, we tried to cut the longer one to match the length of the shorter. This made the accuracy even worse. It seems that the parts cut out from longer documents contain valuable information for LRD. We decided to use the entire strings for LRD, despite the noise brought by the variation of string lengths.

Method	Accuracy
KRR + $K_{nLRD_{6+8}}$	75.4%
KRR + $\hat{k}^{0/1}_{5-8}$ + $K_{nLRD_{6+8}}$	$\mathbf{81.6}\%$
KRR + $(\hat{k}^{0/1} + K_{nLRD})_{6+8}$	80.9%

Table 3: Accuracy rates of different kernel combinations using 10-fold cross-validation on the training set.

4.5 Combining Kernels

To improve results, we thought of combining the kernels in different ways. First, notice that the blended string kernels presented in section 4.3 are essentially a sum of the string kernels with different n-grams. This combination improves the accuracy, being more stable and robust. In the same manner, the LRD kernels based on 6-grams and 8-grams, respectively, were summed up to obtain the kernel denoted by $K_{nLRD_{6+8}}$. Indeed, the $K_{nLRD_{6+8}}$ kernel works better (see Table 3).

There are other options to combine the string kernels with LRD kernels, besides summing them up. One option is by kernel alignment (Cristianini et al., 2001). Instead of simply summing kernels, kernel alignment assigns weights for each to the two kernels based on how well they are aligned with the ideal kernel YY' obtained from labels. Thus, the 5-8-grams presence bits normalized kernel ($\hat{k}^{0/1}_{5-8}$) was combined with the LRD kernel based on sum of 6,8-grams ($K_{nLRD_{6+8}}$), by kernel alignment. From our experiments, kernel alignment worked slightly better than the sum of the two kernels. This also suggests that kernels can be combined only by kernel alignment. The string kernel of length 6 was aligned with the LRD kernel based on 6-grams. In the same way, the string kernel of length 8 was aligned with the LRD kernel based on 8-grams. The two kernels obtained by alignment are combined together, again by kernel alignment, to obtain the kernel denoted by $(\hat{k}^{0/1} + K_{nLRD})_{6+8}$. The results of all kernel combinations are presented in Table 3. The reported accuracy rates were obtained with the KRR parameter

Method	**Submission**	**CV Tr.**	**Dev.**	**CV Tr.+Dev.**	**Test**
KRR + $\hat{k}^{0/1}_{5-8}$	Unibuc-1	80.9%	85.4%	82.5%	82.0%
KRR + $K_{nLRD_{6+8}}$	Unibuc-2	75.4%	76.3%	75.7%	75.8%
KRR + $\hat{k}^{0/1}_{5-8}$ + $K_{nLRD_{6+8}}$	Unibuc-3	$\mathbf{81.6\%}$	$\mathbf{85.7\%}$	$\mathbf{82.6\%}$	82.5%
KRR + $(\hat{k}^{0/1} + K_{nLRD})_{6+8}$	Unibuc-4	80.9%	85.6%	82.0%	81.4%
KRR + $\hat{k}^{0/1}_{5-8}$ + $K_{nLRD_{6+8}}$ + heuristic	Unibuc-5	-	-	-	$\mathbf{82.7\%}$

Table 4: Accuracy rates of submitted systems on different evaluation sets. The Unibuc team ranked third in the closed NLI Shared Task with the kernel combination improved by the heuristic to level the predicted class distribution.

λ set to 10^{-5}.

4.6 Results and Discussion

For the closed NLI Shared Task we submitted the two main systems, namely the 5-8-grams presence bits normalized kernel and the LRD kernel based on sum of 6,8-grams, separately. Another two submissions are the kernel combinations discussed in section 4.5. These four systems were tested using several evaluation procedures, with results shown in Table 4. First, they were tested using 10-fold cross validation on the training set. Next, the systems were tested on the development set. In this case, the systems were trained on the entire training corpus. Another 10-fold cross validation procedure was done on the corpus obtained by combining the training and the development sets. The folds were provided by the organizers. Finally, the results of our systems on the NLI Shared Task test set are given in the last column of Table 4. For testing, the systems were trained on the entire training and development set, with the KRR parameter λ set to $2 \cdot 10^{-5}$.

We didn't expect $K_{nLRD_{6+8}}$ kernel to perform very well on the test set. This system was submitted just to be compared with systems submitted by other participants. Considering that LRD is inspired from biology and that it has no ground in computational linguistics, it performed very well, by standing in the top half of the ranking of all submitted systems.

The kernel obtained by aligning the $\hat{k}^{0/1}_{5-8}$ and $K_{nLRD_{6+8}}$ kernels gives the best results, no matter the evaluation procedure. It is followed closely by the other two submitted systems.

We thought of exploiting the distribution of the testing set in our last submitted system. We knew that there should be exactly 100 examples per class for testing. We took the kernel obtained by combining the $\hat{k}^{0/1}_{5-8}$ and $K_{nLRD_{6+8}}$ kernels, and tried to adjust its output to level the predicted class distribution. We took all the classes with more than 100 examples and ranked the examples by their confidence score (returned by regression) to be part of the predicted class. The examples ranked below 100 were chosen to be redistributed to the classes that had less than 100 examples per class. Examples were redistributed only if their second most confident class had less than 100 examples. This heuristic improved the results on the test set by 0.2%, enough to put us on third place in the closed NLI Shared Task.

5 Conclusion

In this paper, we have presented our approach to the 2013 NLI Shared Task. What makes our system stand out is that it works at the character level, making the approach completely language independent and linguistic theory neutral. The results obtained were very good. A standard approach based on string kernels, that proved to work well in many text analysis tasks, obtained an accuracy of 82% on test data with a difference of only 1.6% between it and the top performing system. A second system based on a new kernel K_{LRD}, inspired from biology with no ground in computational linguistics, performed also unexpectedly well, by standing in the top half of the ranking of all submitted systems. The combination of the two kernels obtained an accuracy of 82.5% making it to the top ten, while an heuristic improvement of this combination ranked third with an accuracy of 82.7%. Obviously, an explanation for these results was needed. It will be adressed in future work.

References

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A Corpus of Non-Native English. Technical report, Educational Testing Service.

Julian Brooke and Graeme Hirst. 2012. Robust, Lexicalized Native Language Identification. In *Proceedings of COLING 2012*, pages 391–408, Mumbai, India, December. The COLING 2012 Organizing Committee.

Serhiy Bykh and Detmar Meurers. 2012. Native Language Identification using Recurring n-grams – Investigating Abstraction and Domain Dependence. In *Proceedings of COLING 2012*, pages 425–440, Mumbai, India, December. The COLING 2012 Organizing Committee.

Markus Chimani, Matthias Woste, and Sebastian Bocker. 2011. A Closer Look at the Closest String and Closest Substring Problem. *Proceedings of ALENEX*, pages 13–24.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. 2001. On kernel-target alignment. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 367–373. MIT Press.

Liviu P. Dinu and Radu Tudor Ionescu. 2012a. An Efficient Rank Based Approach for Closest String and Closest Substring. *PLoS ONE*, 7(6):e37576, 06.

Liviu P. Dinu and Radu Tudor Ionescu. 2012b. Clustering based on Rank Distance with Applications on DNA. *Proceedings of ICONIP*, 7667:722–729.

Liviu P. Dinu and Andrea Sgarro. 2006. A Low-complexity Distance for DNA Strings. *Fundamenta Informaticae*, 73(3):361–372.

Liviu P. Dinu. 2003. On the classification and aggregation of hierarchies with different constitutive elements. *Fundamenta Informaticae*, 55(1):39–50.

C. Grozea, C. Gehl, and M. Popescu. 2009. ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In *3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE*, page 10.

Radu Tudor Ionescu. 2013. Local Rank Distance and its Applications on DNA. Submitted to PKDD.

Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. 2002. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575.

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul M. B. Vitanyi. 2004. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Pall Melsted and Jonathan Pritchard. 2011. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinformatics*, 12(1):333.

Marius Popescu and Liviu P. Dinu. 2007. Kernel methods and string kernels for authorship identification: The federalist papers case. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-07)*, Borovets, Bulgaria, September.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*.

Marius Popescu. 2011. Studying translationese at the character level. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 634–639, Hissar, Bulgaria, September. RANLP 2011 Organising Committee.

V. Yu. Popov. 2007. Multiple genome rearrangement by swaps and by element duplications. *Theoretical Computer Science*, 385(1-3):115–126.

Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(January):101–141.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 482–491, Sydney, Australia, July. Association for Computational Linguistics.

Dana Shapira and James A. Storer. 2003. Large Edit Distance with Multiple Block Operations. *Proceedings of SPIRE*, 2857:369–377.

J. S. Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native Tongues, Lost and Found: Resources and Empirical Evaluations in Native Language Identification. In *Proceedings of COLING 2012*, pages 2585–2602, Mumbai, India, December. The COLING 2012 Organizing Committee.

Francesco Vezzi, Cristian Del Fabbro, Alexandru I. Tomescu, and Alberto Policriti. 2012. rNA: a fast and accurate short reads numerical aligner. *Bioinformatics*, 28(1):123–124.