Can characters reveal your native language? A language-independent approach to native language identification

Radu Tudor Ionescu[◇], **Marius Popescu**[◇], **Aoife Cahill**[†]

[◇]University of Bucharest
Department of Computer Science
14 Academiei, Bucharest, Romania
raducu.ionescu@gmail.com
popescunmarius@gmail.com

[†]Educational Testing Service
660 Rosedale Rd
Princeton, NJ 08541, USA
acahill@ets.org

Abstract

A common approach in text mining tasks such as text categorization, authorship identification or plagiarism detection is to rely on features like words, part-of-speech tags, stems, or some other high-level linguistic features. In this work, an approach that uses character n-grams as features is proposed for the task of native language identification. Instead of doing standard feature selection, the proposed approach combines several string kernels using multiple kernel learning. Kernel Ridge Regression and Kernel Discriminant Analysis are independently used in the learning stage. The empirical results obtained in all the experiments conducted in this work indicate that the proposed approach achieves state of the art performance in native language identification, reaching an accuracy that is 1.7% above the top scoring system of the 2013 NLI Shared Task. Furthermore, the proposed approach has an important advantage in that it is language independent and linguistic theory neutral. In the cross-corpus experiment, the proposed approach shows that it can also be topic independent, improving the state of the art system by 32.3%.

1 Introduction

Using words as basic units is natural in textual analysis tasks such as text categorization, authorship identification or plagiarism detection. Perhaps surprisingly, recent results indicate that methods handling the text at the character level can also be very effective (Lodhi et al., 2002; Sanderson and Guenter, 2006; Popescu and Dinu, 2007; Grozea et al., 2009; Popescu, 2011; Popescu and Grozea, 2012). By disregarding features of natural language such as words, phrases, or meaning, an approach that works at the character level has an important advantage in that it is language independent and linguistic theory neutral. This paper presents a state of the art machine learning system for native language identification that works at the character level. The proposed system is inspired by the system of Popescu and Ionescu (2013), but includes some variations and improvements. A major improvement is that several string kernels are combined via multiple kernel learning (Shawe-Taylor and Cristianini, 2004). Despite the fact that the (histogram) intersection kernel is very popular in computer vision (Maji et al., 2008; Vedaldi and Zisserman, 2010), it has never been used before in text mining. In this work, the intersection kernel is used for the first time in a text categorization task, alone and in combination with other kernels. The intersection kernel lies somewhere in the middle between the kernel that takes into account only the presence of n-grams and the kernel based on the frequency of n-grams (p-spectrum string kernel).

Two kernel classifiers are proposed for the learning task, namely Kernel Ridge Regression (KRR) and Kernel Discriminant Analysis (KDA). The KDA classifier is able to avoid the class-masking problem (Hastie and Tibshirani, 2003), which may often arise in the context of native language identification. Several experiments are conducted to evaluate the performance of the approach proposed in this work. While multiple kernel learning seems to produce a more robust system, the two kernel classifiers obtained mixed results in the experiments. Overall, the empirical results indicate that the approach proposed in this paper achieves state of the art performance in native language identification, while being both lan-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1363–1373,
October 25-29, 2014, Doha, Qatar.

guage independent and linguistic theory neutral. Furthermore, the approach based on string kernels does not need any expert knowledge of words or phrases in the language.

The paper is organized as follows. Related work is presented in Section 2. Section 3 presents several similarity measures for strings, including string kernels and Local Rank Distance. The learning methods used in the experiments are described in Section 4. Section 5 presents details about the experiments. Finally, the conclusions are drawn in Section 6.

2 Related Work

2.1 Native Language Identification

The goal of automatic native language identification (NLI) is to determine the native language of a language learner, based on a piece of writing in a foreign language. This can provide useful information in forensic linguistic tasks (Estival et al., 2007) or could be used in an educational setting to provide contrastive feedback to language learners. Most research has focused on identifying the native language of English language learners, though there have been some efforts recently to identify the native language of writing in other languages (Malmasi and Dras, 2014).

In general most approaches to NLI have used multi-way classification with SVMs or similar models along with a range of linguistic features. The seminal paper by Koppel et al. (2005) introduced some of the best-performing features: character, word and part-of-speech n-grams along with features inspired by the work in the area of second-language acquisition such as spelling and grammatical errors. In 2013, Tetreault et al. (2013) organized the first shared task in the field. This allowed researchers to compare approaches for the first time on a specifically designed NLI corpus that was much larger than previously available data sets. In the shared task, 29 teams submitted results for the test set, and one of the most successful aspects of the competition was that it drew submissions from teams working in a variety of research fields. The submitted systems utilized a wide range of machine learning approaches, combined with several innovative feature contributions. The best performing system achieved an overall accuracy of 83.6% on the 11-way classification of the test set, although there was no significant difference between the top teams.

2.2 Methods that Work at the Character Level

In recent years, methods of handling text at the character level have demonstrated impressive performance levels in various 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). Lodhi et al. (2002) used string kernels for document categorization with very good results. 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.

Using string kernels makes 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 instance, they select only features that reflect various types of spelling errors or only some type of words, such as function words. These features prove to be very effective for specific tasks, but it is possible that other good features also exist. String kernels embed the texts in a very large feature space, given by all the substrings of length p, and leave it to the learning algorithm to select important features for the specific task, by highly weighting these features. It is important to note that this approach is also linguistic theory neutral, since it disregards any features of natural language such as words, phrases, or meaning. On the other hand, a method that considers words as features cannot be completely language independent, since the definition of a word is necessarily language-specific. For example, a method that uses only function words as features is not completely language independent because it needs a list of function words which is specific to a language. When features such as part-of-speech tags are used, as in the work of Jarvis et al. (2013), the method relies on a part-of-speech tagger which might not be available (yet) for some languages. Furthermore, a way to segment a text into words is not an easy task for some languages, such as Chinese.

Character n-grams are used by some of the systems developed for native language identification.

In work where feature ablation results have been reported, the performance with only character n-gram features was modest compared to other types of features (Tetreault et al., 2012). Initially, most work limited the character features to unigrams, bigrams and trigrams, perhaps because longer n-grams were considered too expensive to compute or unlikely to improve performance. However, some of the top systems in the 2013 NLI Shared Task were based on longer character n-grams, up to 9-grams (Jarvis et al., 2013; Popescu and Ionescu, 2013). The results presented in this work are obtained using a range of 5–8 n-grams. Combining all 5–8 n-grams would generate millions of features, which are indeed expensive to compute and represent. The key to avoiding the computation of such a large number of features lies in using the dual representation provided by the string kernel. String kernel similarity matrices can be computed much faster and are extremely useful when the number of samples is much lower than the number of features.

3 Similarity Measures for Strings

3.1 String Kernels

The kernel function gives kernel methods the power to naturally handle input data that is not in the form of numerical vectors, e.g. 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 (Shawe-Taylor and Cristianini, 2004).

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.[1] The feature map defined by this kernel associates a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p (p-grams) with each string.

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

In computer vision, the (histogram) intersection kernel has successfully been used for object class recognition from images (Maji et al., 2008; Vedaldi and Zisserman, 2010). In this paper, the intersection kernel is used for the first time as a kernel for strings. The intersection string kernel is defined as follows:

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

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s.

For the p-spectrum kernel, the frequency of a p-gram has a very significant contribution to the kernel, since it considers the product of such frequencies. On the other hand, the frequency of a p-gram is completely disregarded in the p-grams presence bits kernel. The intersection kernel lies somewhere in the middle between the p-grams presence bits kernel and p-spectrum kernel, in the sense that the frequency of a p-gram has a moderate contribution to the intersection kernel. More precisely, the following inequality that describes the relation between the three kernels holds:

$$k_p^{0/1}(s,t) \leq k_p^{\cap}(s,t) \leq k_p(s,t).$$

What is actually more interesting is that the intersection kernel assigns a high score to a p-gram if it has a high frequency in both strings, since it considers the minimum of the two frequencies. The p-spectrum kernel assigns a high score even when the p-gram has a high frequency in only one of the two strings. Thus, the intersection kernel captures something about the correlation between the p-gram frequencies in the two strings, which may lead to a more sensitive similarity between strings.

[1] Note that the notion of substring requires contiguity. Shawe-Taylor and Cristianini (2004) discuss the ambiguity between the terms *substring* and *subsequence* across different domains: biology, computer science.

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

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

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

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

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

The string kernel implicitly embeds the texts in a high dimensional feature space. Then, a kernel-based learning algorithm implicitly assigns a weight to each feature, thus selecting the features that are important for the discrimination task. For example, in the case of text categorization the learning algorithm enhances the features representing stems of content words (Lodhi et al., 2002), while in the case of authorship identification the same learning algorithm enhances the features representing function words (Popescu and Dinu, 2007).

3.2 Local Rank Distance

A recently introduced distance measure, termed Local Rank Distance (Ionescu, 2013), comes from the idea of better adapting rank distance (Dinu, 2003) to string data, in order to capture a better similarity between strings, such as DNA sequences or text. Local Rank Distance (LRD) has already shown promising results in computational biology (Ionescu, 2013) and native language identification (Popescu and Ionescu, 2013).

In order to describe LRD, the following notations are defined. Given a string x over an alphabet Σ, and a character $a \in \Sigma$, the length of x is denoted by $|x|$. Strings are considered to be indexed starting from position 1, that is $x = x[1]x[2]\cdots x[|x|]$. Moreover, $x[i:j]$ denotes its substring $x[i]x[i+1]\cdots x[j-1]$.

Local Rank Distance is inspired by rank distance (Dinu, 2003), the main differences being that it uses p-grams instead of single characters, and that it matches each p-gram in the first string with the nearest equal p-gram in the second string. Given a fixed integer $p \geq 1$, a threshold $m \geq 1$, and two strings x and y over Σ, the *Local Rank Distance* between x and y, denoted by $\Delta_{LRD}(x,y)$, is defined through the following algorithmic process. For each position i in x ($1 \leq i \leq |x|-p+1$), the algorithm searches for that position j in y ($1 \leq j \leq |y|-p+1$) such that $x[i:i+p] = y[j:j+p]$ and $|i-j|$ is minimized. If j exists and $|i-j| < m$, then the offset $|i-j|$ is added to the Local Rank Distance. Otherwise, the maximal offset m is added to the Local Rank Distance. An important remark is that LRD does not impose any mathematically developed global constraints, such as matching the i-th occurrence of a p-gram in x with the i-th occurrence of that same p-gram in y. Instead, it is focused on the local phenomenon, and tries to pair equal p-grams at a minimum offset. To ensure that LRD is a (symmetric) distance function, the algorithm also has to sum up the offsets obtained from the above process by exchanging x and y. LRD can be formally defined as follows.

Definition 1 *Let $x, y \in \Sigma^*$ be two strings, and let $p \geq 1$ and $m \geq 1$ be two fixed integer values. The Local Rank Distance between x and y is defined as:*

$$\Delta_{LRD}(x,y) = \Delta_{left}(x,y) + \Delta_{right}(x,y),$$

where $\Delta_{left}(x,y)$ and $\Delta_{right}(x,y)$ are defined as follows:

$$\Delta_{left}(x,y) = \sum_{i=1}^{|x|-p+1} \min\{|i-j| \text{ such that } 1 \leq j \leq |y|-p+1 \text{ and } x[i:i+p] = y[j:j+p]\} \cup \{m\},$$

$$\Delta_{right}(x,y) = \sum_{j=1}^{|y|-p+1} \min\{|j-i| \text{ such that } 1 \leq i \leq |x|-p+1 \text{ and } y[j:j+p] = x[i:i+p]\} \cup \{m\}.$$

Interestingly, the search for matching p-grams is limited within a window of fixed size. The size of this window is determined by the maximum offset parameter m. This parameter must be set a priori and should be proportional to the size of the alphabet, the p-grams, and to the lengths of the strings.

The following example offers a better understanding of how LRD actually works. LRD is computed between two strings using 2-grams.

Example 1 *Given two strings $x = abcaa$ and $y = cabca$, a fixed maximal offset $m = 3$, and*

a fixed size of p-grams $p = 2$, Δ_{left} and Δ_{right} are computed as follows:

$$\begin{aligned}\Delta_{left}(x, y) &= |1 - 2| + |2 - 3| \\ &\quad + |3 - 4| + 3 = 6, \\ \Delta_{right}(x, y) &= |1 - 3| + |2 - 1| \\ &\quad + |3 - 2| + |4 - 3| = 5.\end{aligned}$$

By summing up the two partial sums, Local Rank Distance is obtained

$$\Delta_{LRD}(x, y) = \Delta_{left}(x, y) + \Delta_{right}(x, y) = 11.$$

The maximum LRD value between two strings can be computed as the product between the maximum offset m and the number of pairs of compared p-grams. Thus, LRD can be normalized to a value in the $[0, 1]$ interval. By normalizing, LRD becomes 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 (Shawe-Taylor and Cristianini, 2004):

$$\hat{k}_p^{LRD}(s, t) = e^{-\frac{\Delta_{LRD}(s,t)}{2\sigma^2}},$$

where s and t are two strings and p is the p-grams length. The parameter σ is usually chosen so that values of $\hat{k}(s, t)$ are well scaled. In the above equation, Δ_{LRD} is already normalized to a value in the $[0, 1]$ interval to ensure a fair comparison of strings of different length.

4 Learning Methods

Kernel-based learning algorithms work by embedding the data into a Hilbert feature 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. More precisely, a kernel matrix that contains the pairwise similarities between every pair of training samples is used in the learning stage to assign a vector of weights to the training samples. Let α denote this weight vector. In the test stage, the pairwise similarities between a test sample x and all the training samples are computed. Then, the following binary classification function assigns a positive or a negative label to the test sample:

$$g(x) = \sum_{i=1}^{n} \alpha_i \cdot k(x, x_i),$$

where x is the test sample, n is the number of training samples, $X = \{x_1, x_2, ..., x_n\}$ is the set of training samples, k is a kernel function, and α_i is the weight assigned to the training sample x_i. In the primal form, the same binary classification function can be expressed as:

$$g(x) = \langle w, x \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product, $x \in \mathbb{R}^m$ is the test sample represented as a vector of features, and $w \in \mathbb{R}^m$ is a vector of feature weights that can be computed as follows:

$$w = \sum_{i=1}^{n} \alpha_i \cdot x_i,$$

given that the kernel function k can be expressed as a scalar product between samples.

The advantage of using the dual representation induced by the kernel function becomes clear if the dimension of the feature space m is taken into consideration. Since string kernels are based on character n-grams, the feature space is indeed very high. For instance, using 5-grams based only on the 26 letters of the English alphabet will result in a feature space of $26^5 = 11,881,376$ features. However, in the experiments presented in this work the feature space includes 5-grams along with 6-grams, 7-grams and 8-grams. As long as the number of samples n is not greater than the number of features m, it is more efficient to use the dual representation given by the kernel matrix. This fact is also known as the *kernel trick* (Shawe-Taylor and Cristianini, 2004).

Various kernel methods differ in the way they learn to separate the samples. 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. For the NLI experiments, two binary kernel classifiers are used, namely the SVM (Cortes and Vapnik, 1995), and the KRR. Support Vector Machines try to find the vector of weights that defines the hyperplane that maximally separates the images in the Hilbert space of the training examples

belonging to the two classes. Kernel Ridge Regression selects the vector of weights that simultaneously has small empirical error and small norm in the Reproducing Kernel Hilbert Space generated by the kernel function. More details about SVM and KRR can be found in (Shawe-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.

SVM and KRR produce binary classifiers, but native language identification is usually a multi-class classification problem. There are many approaches for combining binary classifiers to solve multi-class problems. Typically, the multi-class problem is broken down into multiple binary classification problems using common decomposing schemes such as: one-versus-all and one-versus-one. There are also kernel methods that take the multi-class nature of the problem directly into account, e.g. Kernel Discriminant Analysis. The KDA classifier is able to improve accuracy by avoiding the masking problem (Hastie and Tibshirani, 2003). In the case of multi-class native language identification, the masking problem may appear when non-native English speakers have acquired, as the second language, a different language to English. For example, an essay written in English produced by a French native speaker that is also proficient in German, could be identified as either French or German.

5 Experiments

5.1 Data Sets Description

In this paper, experiments are carried out on three datasets: a modified version of the ICLEv2 corpus (Granger et al., 2009), the ETS Corpus of Non-Native Written English, or TOEFL11 (Blanchard et al., 2013), and the TOEFL11-Big corpus as used by Tetreault et al. (2012). A summary of the corpora is given in Table 1.

Corpus	Languages	Documents
ICLE	7	770
TOEFL11	11	12,100
TOEFL11-Big	11	87,502

Table 1: Summary of corpora used in the experiments.

The ICLEv2 is a corpus of essays written by highly-proficient non-native college-level students of English. For many years this was the standard corpus used in the task of native language identification. However, the corpus was originally collected for the purpose of corpus linguistic investigations, and because of this contains some idiosyncrasies that make it problematic for the task of NLI (Brooke and Hirst, 2012). Therefore, a modified version of the corpus that has been normalized as much as possible for topic and character encoding (Tetreault et al., 2012) is used. This version of the corpus contains 110 essays each for 7 native languages: Bulgarian, Chinese, Czech, French, Japanese, Russian and Spanish.

The ETS Corpus of Non-Native Written English (TOEFL11) was first introduced by Tetreault et al. (2012) and extended for the 2013 Native Language Identification Shared Task (Tetreault et al., 2013). It was designed to overcome many of the shortcomings identified with using the ICLEv2 corpus for this task. The TOEFL11 corpus contains a balanced distribution of essays per prompt (topic) per native language. It also contains information about the language proficiency of each writer. The corpus contains essays written by speakers of the following 11 languages: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu and Turkish. For the shared task, the 12,100 essays were split into 9,900 for training, 1,100 for development and 1,100 for testing.

Tetreault et al. (2012) present a corpus, TOEFL11-Big, to investigate the performance of their NLI system on a very large data set. This data set contains the same languages as TOEFL11, but with no overlap in content. It contains a total of over 87 thousand essays written to a total of 76 different prompts. The distribution of L1 per prompt is not as even as for TOEFL11, though all topics are represented for all L1s.

5.2 Parameter Tuning and Implementation Choices

In the string kernels approach proposed in this work, documents or essays from this corpus are treated as strings. Therefore, the notions of *string* or *document* is used interchangeably throughout this work. Because the approach works at the character level, there is no 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, and so on) with a single space character. This normalization was needed in order to prevent the artificial increase or decrease of the similarity between texts, as a result of different spacing. All uppercase letters were converted to the corresponding lowercase ones.

A series of preliminary experiments were conducted in order to select the best-performing learning method. In these experiments the string kernel was fixed to the p-spectrum normalized kernel of length 5 ($\hat{k}_5$), because the goal was to select the best learning method, and not to find the best kernel. The following learning methods were evaluated: one-versus-one SVM, one-versus-all SVM, one-versus-one KRR, one-versus-all KRR, and KDA. A 10-fold cross-validation procedure was carried out on the TOEFL11 training set to evaluate the classifiers. The preliminary results indicate that the one-versus-all KRR and the KDA classifiers produce the best results. Therefore, they are selected for the remaining experiments.

Another set of preliminary experiments were performed to determine the range of n-grams that gives the most accurate results on a 10-fold cross-validation procedure carried out on the TOEFL11 training set. All the n-grams in the range 2-10 were evaluated. Furthermore, experiments with different blended kernels were conducted to see whether combining n-grams of different lengths could improve the accuracy. The best results were obtained when all the n-grams with the length in the range 5-8 were used. Other authors (Bykh and Meurers, 2012; Popescu and Ionescu, 2013) also report better results by using n-grams with the length in a range, rather than using n-grams of fixed length. Consequently, the results reported in this work are based on blended string kernels based on 5-8 n-grams.

Some preliminary experiments were also performed to establish the type of kernel to be used, namely the blended p-spectrum kernel ($\hat{k}_{5-8}$), the blended p-grams presence bits kernel ($\hat{k}^{0/1}_{5-8}$), the blended p-grams intersection kernel ($\hat{k}^{\cap}_{5-8}$), or the kernel based on LRD ($\hat{k}^{LRD}_{5-8}$.). These different kernel representations are obtained from the same data. The idea of combining all these kernels is natural when one wants to improve the performance of a classifier. When multiple kernels are combined, the features are actually embedded in a higher-dimensional space. As a consequence, the search space of linear patterns grows, which helps the classifier to select a better discriminant function. The most natural way of combining two kernels is to sum them up. Summing up kernels or kernel matrices is equivalent to feature vector concatenation. Another option is to combine kernels by kernel alignment (Cristianini et al., 2001). Instead of simply summing kernels, kernel alignment assigns weights for each of the two kernels based on how well they are aligned with the ideal kernel YY' obtained from training labels. The kernels were evaluated alone and in various combinations. The best kernels are the blended p-grams presence bits kernel and the blended p-grams intersection kernel. The best kernel combinations include the blended p-grams presence bits kernel, the blended p-grams intersection kernel and the kernel based on LRD. Since the kernel based on LRD is slightly slower than the other string kernels, the kernel combinations that include it were only evaluated on the TOEFL11 corpus and on the ICLE corpus.

5.3 Experiment on TOEFL11 Corpus

This section describes the results on the TOEFL11 corpus. Thus, results for the 2013 *Closed* NLI Shared Task are also included. In the closed shared task the goal is to predict the native language of testing examples, restricted to learning only from the training and the development data. The additional information from *prompts* or the English language proficiency level were not used in the proposed approach.

The regularization parameters were tuned on the development set. In this case, the systems were trained on the entire training set. A 10-fold cross-validation (CV) procedure was done on the training and the development sets. The folds were provided along with the TOEFL11 corpus. Finally, the results of the proposed systems are also reported on the NLI Shared Task test set. For testing, the systems were trained on both the training set and the development set. The results are summarized in Table 2.

The results presented in Table 2 show that string kernels can reach state of the art accuracy levels for this task. Overall, it seems that KDA is able to obtain better results than KRR. The intersection kernel alone is able to obtain slightly better results than the presence bits kernel. The kernel based on LRD gives significantly lower accuracy rates, but it is able to improve the performance when it is

Method	Development	10-fold CV	Test
Ensemble model (Tetreault et al., 2012)	-	80.9%	-
KRR and string kernels (Popescu and Ionescu, 2013)	-	82.6%	82.7%
SVM and word features (Jarvis et al., 2013)	-	**84.5%**	83.6%
KRR and $\hat{k}^{0/1}_{5-8}$	85.4%	82.5%	82.0%
KRR and $\hat{k}^{\cap}_{5-8}$	84.9%	82.2%	82.6%
KRR and $\hat{k}^{LRD}_{5-8}$	78.7%	77.1%	77.5%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	85.7%	82.6%	82.7%
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	84.9%	82.2%	82.0%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	85.5%	82.6%	82.5%
KRR and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	85.5%	82.6%	82.5%
KDA and $\hat{k}^{0/1}_{5-8}$	86.2%	83.6%	83.6%
KDA and $\hat{k}^{\cap}_{5-8}$	85.2%	83.5%	84.6%
KDA and $\hat{k}^{LRD}_{5-8}$	79.7%	78.5%	79.2%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	**87.1%**	84.0%	84.7%
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	85.8%	83.4%	83.9%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	86.4%	84.1%	85.0%
KDA and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	86.5%	84.1%	**85.3%**
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	87.0%	84.1%	84.8%

Table 2: Accuracy rates on TOEFL11 corpus of various classification systems based on string kernels compared with other state of the art approaches. The best accuracy rates on each set of experiments are highlighted in bold. The weights a_1 and a_2 from the weighted sums of kernels are computed by kernel alignment.

combined with the blended p-grams presence bits kernel. In fact, most of the kernel combinations give better results than each of their components. The best kernel combination is that of the presence bits kernel and the intersection kernel. Results are quite similar when they are combined either by summing them up or by kernel alignment. The best performance on the test set (85.3%) is obtained by the system that combines these two kernels via kernel alignment and learns using KDA. This system is 1.7% better than the state of the art system of Jarvis et al. (2013) based on SVM and word features, this being the top scoring system in the NLI 2013 Shared Task. It is also 2.6% better than the state of the art system based on string kernels of Popescu and Ionescu (2013). On the cross validation procedure, there are three systems that reach the accuracy rate of 84.1%. All of them are based on KDA and various kernel combinations. The greatest accuracy rate of 84.1% reported for the cross validation procedure is 3.2% above the state of the art system of Tetreault et al. (2012) and 0.4% below the top scoring system of Jarvis et al. (2013). The empirical results obtained in this experiment demonstrate that the approach proposed in this paper can reach state of the art accuracy levels. It is worth mentioning that a significance test performed by the organizers of the NLI 2013 Shared Task showed that the top systems that participated in the competition are not essentially different. Further experiments on the ICLE corpus and on the TOEFL11-Big corpus are conducted to determine whether the approach proposed in this paper is significantly better than other state of the art approaches.

5.4 Experiment on ICLE Corpus

The results on the ICLE corpus using a 5-fold cross validation procedure are summarized in Table 3. To adequately compare the results with a state of the art system, the same 5-fold cross validation procedure used by Tetreault et al. (2012) was also used in this experiment. Table 3 shows that the results obtained by the presence bits kernel and by the intersection kernel are systematically better than the state of the art system of Tetreault et al. (2012). While both KRR and KDA produce accuracy rates that are better than the state of the art accuracy rate, it seems that KRR is slightly better in this experiment. Again, the idea of combining kernels seems to produce more robust systems. The best systems are based on combining the presence bits kernel either with the kernel based on LRD or the intersection kernel. Overall, the reported accuracy rates are higher than the state of the art accuracy rate. The best performance (91.3%) is achieved by the KRR classifier based on combining the presence bits kernel with

Method	5-fold CV
Ensemble model (Tetreault et al., 2012)	90.1%
KRR and $\hat{k}^{0/1}_{5-8}$	91.2%
KRR and $\hat{k}^{\cap}_{5-8}$	90.5%
KRR and $\hat{k}^{LRD}_{5-8}$	81.8%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	**91.3%**
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.1%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	90.9%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.6%
KDA and $\hat{k}^{0/1}_{5-8}$	90.5%
KDA and $\hat{k}^{\cap}_{5-8}$	90.5%
KDA and $\hat{k}^{LRD}_{5-8}$	82.3%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.8%
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.4%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	91.0%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	90.8%

Table 3: Accuracy rates on ICLE corpus of various classification systems based on string kernels compared with a state of the art approach. The accuracy rates are reported for the same 5-fold CV procedure as in (Tetreault et al., 2012). The best accuracy rate is highlighted in bold.

the kernel based on LRD. This represents an 1.2% improvement over the state of the art accuracy rate of Tetreault et al. (2012). Two more systems are able to obtain accuracy rates greater than 91.0%. These are the KRR classifier based on the presence bits kernel (91.2%) and the KDA classifier based on the sum of the presence bits kernel and the intersection kernel (91.0%). The overall results on the ICLE corpus show that the string kernels approach can reach state of the art accuracy levels. It is worth mentioning the purpose of this experiment was to use the same approach determined to work well in the TOEFL11 corpus. To serve this purpose, the range of n-grams was not tuned on this data set. Furthermore, other classifiers were not tested in this experiment. Nevertheless, better results can probably be obtained by adding these aspects into the equation.

5.5 Cross-corpus Experiment

In this experiment, various systems based on KRR or KDA are trained on the TOEFL11 corpus and tested on the TOEFL11-Big corpus. The kernel based on LRD was not included in this experiment since it is more computationally expensive. Therefore, only the presence bits kernel and the intersection kernel were evaluated on the TOEFL11-Big corpus. The results are summarized in Table 4. The same regularization parameters determined to work well on the TOEFL11 development set were used.

Method	Test
Ensemble model (Tetreault et al., 2012)	35.4%
KRR and $\hat{k}^{0/1}_{5-8}$	66.7%
KRR and $\hat{k}^{\cap}_{5-8}$	67.2%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	**67.7%**
KRR and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	**67.7%**
KDA and $\hat{k}^{0/1}_{5-8}$	65.6%
KDA and $\hat{k}^{\cap}_{5-8}$	65.7%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	66.2%
KDA and $a_1\hat{k}^{0/1}_{5-8} + a_2\hat{k}^{\cap}_{5-8}$	66.2%

Table 4: Accuracy rates on TOEFL11-Big corpus of various classification systems based on string kernels compared with a state of the art approach. The systems are trained on the TOEFL11 corpus and tested on the TOEFL11-Big corpus. The best accuracy rate is highlighted in bold. The weights a_1 and a_2 from the weighted sums of kernels are computed by kernel alignment.

The most interesting fact is that all the proposed systems are at least 30% better than the state of the art system. Considering that the TOEFL11-Big corpus contains 87 thousand samples, the 30% improvement is significant without any doubt. Diving into details, it can be observed that the results obtained by KRR are higher than those obtained by KDA. However, both methods perform very well compared to the state of the art. Again, kernel combinations are better than each of their individual kernels alone.

It is important to mention that the significant performance increase is not due to the learning method (KRR or KDA), but rather due to the string kernels that work at the character level. It is not only the case that string kernels are language independent, but for the same reasons they can also be topic independent. Since the topics (prompts) from TOEFL11 are different from the topics from TOEFL11-Big, it becomes clear that a method that uses words as features is strongly affected, since the distribution of words per topic can be completely different. But mistakes that reveal the native language can be captured by character n-grams that can appear more often even in different topics. The results indicate that this is also the case of the approach based on string kernels, which seems to be more robust to such topic variations of the data set. The best system has an accuracy rate that is 32.3% better than the state of

the art system of Tetreault et al. (2012). Overall, the empirical results indicate that the string kernels approach can achieve significantly better results than other state of the art approaches.

6 Conclusions

A language-independent approach to native language identification was presented in this paper. The system works at the character level, making the approach completely language independent and linguistic theory neutral. The results obtained in all the three experiments were very good. The best system presented in this work is based on combining the intersection and the presence string kernels by kernel alignment and on deciding the class label either with KDA or KRR. The best system is 1.7% above the top scoring system of the 2013 NLI Shared Task. Furthermore, it has an impressive generalization capacity, achieving results that are 30% higher than the state of the art method in the cross-corpus experiment.

Despite the fact that the approach based on string kernels performed so well, it remains to be further investigated why this is the case and why such a simple approach can compete with far more complex approaches that take words, lemmas, syntactic information, or even semantics into account. It seems that there are generalizations to the kinds of mistakes that certain non-native English speakers make that can be captured by n-grams of different lengths. Interestingly, using a range of n-grams generates a large number of features including (but not limited to) stop words, stems of content words, word suffixes, entire words, and even n-grams of short words. Rather than doing feature selection before the training step, which is the usual NLP approach, the kernel classifier selects the most relevant features during training. With enough training samples, the kernel classifier does a better job of selecting the right features from a very high feature space. This may be one reason for why the string kernel approach works so well. To gain additional insights into why this technique is working well, the features selected by the classifier as being more discriminating can be analyzed in future work. This analysis would also offer some information about localized language transfer effects, since the features used by the proposed model are n-grams of lengths 5 to 8. As mentioned before, the features captured by the model typically include stems, function words, word prefixes and suffixes, which have the potential to generalize over purely word-based features. These features would offer insights into two kinds of language transfer effects, namely word choice (lexical transfer) and morphological differences.

Acknowledgments

The authors would like to thank Beata Beigman Klebanov, Nitin Madnani and Xinhao Wang from ETS for their helpful comments and suggestions. The author also thank the anonymous reviewers for their valuable insights which lead to improvements in the presentation of this 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 Research Report No. RR–13–24.

Julian Brooke and Graeme Hirst. 2012. Robust, Lexicalized Native Language Identification. *Proceedings of COLING 2012*, pages 391–408, December.

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. 2001. On kernel-target alignment. *Proceedings of NIPS*, pages 367–373, December.

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

Dominique Estival, Tanja Gaustad, Son-Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. *Proceedings of PACLING*, pages 263–272.

Sylviane Granger, Estelle Dagneaux, and Fanny Meunier. 2009. *The International Corpus of Learner English: Handbook and CD-ROM, version 2*. Presses Universitaires de Louvain, Louvain-la-Neuve, Belgium.

Cristian Grozea, Christian Gehl, and Marius 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.

Trevor Hastie and Robert Tibshirani. 2003. *The Elements of Statistical Learning*. Springer, corrected edition, July.

Radu Tudor Ionescu. 2013. Local Rank Distance. *Proceedings of SYNASC*, pages 221–228.

Scott Jarvis, Yves Bestgen, and Steve Pepper. 2013. Maximizing classification accuracy in native language identification. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 111–118, June.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically Determining an Anonymous Author's Native Language. *Proceedings of ISI*, pages 209–217.

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.

Subhransu Maji, Alexander C. Berg, and Jitendra Malik. 2008. Classification using intersection kernel support vector machines is efficient. *Proceedings of CVPR*.

Shervin Malmasi and Mark Dras. 2014. Chinese Native Language Identification. *Proceedings of EACL*, 2:95–99, April.

Marius Popescu and Liviu P. Dinu. 2007. Kernel methods and string kernels for authorship identification: The federalist papers case. *Proceedings of RANLP*, September.

Marius Popescu and Cristian Grozea. 2012. Kernel methods and string kernels for authorship analysis. *CLEF (Online Working Notes/Labs/Workshop)*, September.

Marius Popescu and Radu Tudor Ionescu. 2013. The Story of the Characters, the DNA and the Native Language. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278, June.

Marius Popescu. 2011. Studying translationese at the character level. *Proceedings of RANLP*, pages 634–639, September.

Conrad Sanderson and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. *Proceedings of EMNLP*, pages 482–491, July.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

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

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, June.

Andrea Vedaldi and Andrew Zisserman. 2010. Efficient additive kernels via explicit feature maps. *Proceedings of CVPR*, pages 3539–3546.