String Kernels for Native Language Identification: Insights from Behind the Curtains

Radu Tudor Ionescu*
University of Bucharest

Marius Popescu**
University of Bucharest

Aoife Cahill†
Educational Testing Service

The most common approach in text mining classification tasks is to rely on features like words, part-of-speech tags, stems, or some other high-level linguistic features. Recently, an approach that uses only character p-grams as features has been proposed for the task of native language identification (NLI). The approach obtained state-of-the-art results by combining several string kernels using multiple kernel learning. Despite the fact that the approach based on string kernels performs so well, several questions about this method remain unanswered. First, it is not clear why such a simple approach can compete with far more complex approaches that take words, lemmas, syntactic information, or even semantics into account. Second, although the approach is designed to be language independent, all experiments to date have been on English. This work is an extensive study that aims to systematically present the string kernel approach and to clarify the open questions mentioned above.

A broad set of native language identification experiments were conducted to compare the string kernels approach with other state-of-the-art methods. The empirical results obtained in all of the experiments conducted in this work indicate that the proposed approach achieves state-of-the-art performance in NLI, reaching an accuracy that is 1.7% *above the top scoring system of the 2013 NLI Shared Task. Furthermore, the results obtained on both the Arabic and the Norwegian corpora demonstrate that the proposed approach is language independent. In the Arabic native language identification task, string kernels show an increase of more than* 17% *over the best accuracy reported so far. The results of string kernels on Norwegian native language identification are also significantly better than the state-of-the-art approach. In addition, in a cross-corpus experiment, the proposed approach shows that it can also be topic independent, improving the state-of-the-art system by* 32.3%.

* University of Bucharest, Department of Computer Science, 14 Academiei, Bucharest, Romania. E-mail: raducu.ionescu@gmail.com.
** University of Bucharest, Department of Computer Science, 14 Academiei, Bucharest, Romania. E-mail: popescunmarius@gmail.com.
† Educational Testing Service, 660 Rosedale Rd., Princeton, NJ 08541, USA. E-mail: acahill@ets.org.

Submission received: 6 April 2015; revision received: 23 November 2015; accepted for publication: 5 March 2016.

doi:10.1162/COLI_a_00256

To gain additional insights about the string kernels approach, the features selected by the classifier as being more discriminating are analyzed in this work. The analysis also offers information about localized language transfer effects, since the features used by the proposed model are p-grams of various lengths. The features captured by the model typically include stems, function words, and word prefixes and suffixes, which have the potential to generalize over purely word-based features. By analyzing the discriminating features, this article offers insights into two kinds of language transfer effects, namely, word choice (lexical transfer) and morphological differences. The goal of the current study is to give a full view of the string kernels approach and shed some light on why this approach works so well.

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; Kate and Mooney 2006; Sanderson and Guenter 2006; Popescu and Dinu 2007; Grozea, Gehl, and Popescu 2009; Escalante, Solorio, and Montes-y-Gómez 2011; Popescu 2011; Popescu and Grozea 2012). By avoiding to explicitly consider 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. In this context, a state-of-the-art machine learning system for native language identification (NLI) that works at the character level is presented in this article.

NLI is the task of identifying the native language of a writer, based on a text they have written in a language other than their mother tongue. This is an interesting sub-task in forensic linguistic applications such as plagiarism detection and authorship identification, where the native language of an author is just one piece of the puzzle (Estival et al. 2007). NLI can also play a key role in second language acquisition applications where NLI techniques are used to identify language transfer patterns that help teachers and students focus feedback and learning on particular areas of interest (Rozovskaya and Roth 2010; Jarvis and Crossley 2012).

The system based on string kernels described in this article was initially designed to participate in the 2013 NLI Shared Task (Tetreault, Blanchard, and Cahill 2013) and obtained third place (Popescu and Ionescu 2013). The approach was further extended by Ionescu, Popescu, and Cahill (2014) to combine several string kernels via multiple kernel learning (MKL) (Gonen and Alpaydin 2011). The system revealed some interesting facts about the kinds of string kernels and kernel learning methods that worked well for NLI. For instance, one of the best performing kernels is the (histogram) intersection kernel, which is inspired from computer vision (Maji, Berg, and Malik 2008). Another kernel based on Local Rank Distance is inspired from biology (Ionescu 2013; Dinu, Ionescu, and Tomescu 2014). Two kernel classifiers are alternatively used for the learning task, namely, kernel ridge regression (KRR) and kernel discriminant analysis (KDA). Interestingly, these kernel classifiers give better results than the widely used support vector machines (SVM). Ionescu, Popescu, and Cahill (2014) conducted several experiments to demonstrate state-of-the-art performance of the system based on string kernels. However, the system was only used for the task of identifying the native language of a person from text written in English. Furthermore, a study about the language transfer patterns captured

by the system was not given. The current work aims to clarify these points and to give an extended presentation and evaluation of the string kernels approach.

The first goal of this work is to demonstrate that the system is indeed language-independent, as claimed in Ionescu, Popescu, and Cahill (2014), and to show that it can achieve state-of-the-art performance on a consistent basis. To fulfill this goal, the system is evaluated on several corpora of text documents written in three different languages, namely, Arabic, English, and Norwegian. Having a general character-based approach that works well across languages could be useful, for example, in a streamlined intelligence application that is required to work efficiently on a wide range of languages for which NLP tools or language experts might not be readily available, or where the user does not desire a complex customization for each language. On the other hand, kernels based on a different kind of information (for example, syntactic and semantic information) can be combined with string kernels via MKL to improve accuracy in some specific situations. Indeed, others (Tetreault et al. 2012; Bykh and Meurers 2014; Malmasi and Dras 2014a) have obtained better NLI results by using ensemble models based on several features such as context-free grammar production rules, Stanford dependencies, function words, and part-of-speech p-grams, than using models based on each of these features alone. Combing string kernels with other kind of information is not covered in the present work.

The second goal of this work is to investigate which properties of the simple kernel-based approach are the main driver behind the higher performance than more complex approaches that take words, lemmas, syntactic information, or even semantics into account. The language transfer analysis provided in Section 6 shows that there are generalizations to the kinds of mistakes that certain non-native speakers make that can be captured by p-grams of different lengths. Interestingly, using a range of p-grams implicitly generates a very large number of features including (but not limited to) stop words, stems of content words, word suffixes, entire words, and even p-grams of short words. Rather than doing feature selection before the training step, which is the usual NLP approach, the kernel classifier, aided by regularization, 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 works so well, the features selected by the classifier as being most discriminating are analyzed in this work. The analysis of the discriminant features also provides some information (useful in the context of SLA) about localized language transfer effects, namely, word choice (lexical transfer) and morphological differences.

The article is organized as follows. Work related to native language identification and to methods that work at the character level 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. Section 6 discusses the discriminant features of the string kernels approach and the observed language transfer effects. Finally, conclusions are drawn in Section 7.

2. Related Work

2.1 Native Language Identification

As defined in the Introduction, the goal of automatic NLI is to determine the native language of a language learner, based on a piece of writing in a foreign language. 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, such as Chinese (Malmasi and Dras 2014b) or Arabic (Malmasi and Dras 2014a).

The first work to study automated NLI was that of Tomokiyo and Jones (2001). In their study, a naive Bayes model is trained to distinguish speech transcripts produced by native versus non-native English speakers. A few years later, a second study on NLI appeared (Jarvis, Castañeda Jiménez, and Nielsen 2004). In their work, Jarvis, Castañeda Jiménez, and Nielsen (2004) tried to determine how well a discriminant analysis classifier could predict the L1 language of nearly 500 English learners from different backgrounds. To make the task more challenging, they included pairs of closely related L1 languages, such as Portuguese and Spanish. The seminal paper by Koppel, Schler, and Zigdon (2005) introduced some of the best-performing features for the NLI task: character, word and part-of-speech *p*-grams along with features inspired by the work in the area of second language acquisition such as spelling and grammatical errors. In general, most approaches to NLI have used multi-way classification with SVM or similar models along with a range of linguistic features. The book of Jarvis and Crossley (2012) presents some of the state-of-the-art approaches used up until 2012. Being the first book of its kind, it focuses on the automated detection of L2 language-use patterns that are specific to different L1 backgrounds, with the help of text classification methods. Additionally, the book presents methodological tools to empirically test language transfer hypotheses, with the aim of explaining how the languages that a person knows interact in the mind.

In 2013, Tetreault, Blanchard, and Cahill (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 in the closed task 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. Since the 2013 NLI Shared Task, two more systems have reported results better than the top scoring system of the NLI Shared Task. Bykh and Meurers (2014) presented an ensemble classifier based on lexicalized and non-lexicalized local syntactic features. They explored the context-free grammar (CFG) production rules as syntactic features for the task of NLI. Currently, the state-of-the-art NLI system for this data set is that of Ionescu, Popescu, and Cahill (2014). It is based on a wide range of character *p*-grams. A full presentation of this state-of-the-art system is given in this article.

Another interesting linguistic interpretation of native language identification data was only recently addressed, specifically, the analysis of second language usage patterns caused by native language interference. Usually, language transfer is studied by second language acquisition researchers using manual tools. Language transfer analysis based on automated native language identification methods has been the approach of Jarvis and Crossley (2012). Swanson and Charniak (2014) also define a computational methodology that produces a ranked list of syntactic patterns that are correlated with language transfer. Their methodology allows the detection of fairly obvious language transfer effects, without being able to detect underused patterns. The first work to address the automatic extraction of underused and overused features on

a per-native language basis is that of Malmasi and Dras (2014c). Further similar studies are likely to appear in the years to come. The current work also addresses the automatic extraction of underused and overused features captured by character p-grams. In a similar manner to Malmasi and Dras (2014c), the discriminant features learned by a one-versus-all kernel classifier are analyzed.

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; Kate and Mooney 2006; Popescu and Dinu 2007; Grozea, Gehl, and Popescu 2009; Escalante, Solorio, and Montes-y-Gómez 2011; Popescu 2011; Popescu and Grozea 2012). String kernels are a common form of using information at the character level. They are a particular case of the more general convolution kernels (Haussler 1999). 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 by 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. Because it does not restrict the feature space according to any linguistic theory, the string kernels approach is linguistic theory neutral. Furthermore, the method does not explicitly consider any features of natural language such as words, phrases, or meaning, contrary to the usual NLP approach. However, an interesting remark is that such features can implicitly be discovered within the extracted p-grams. On the other hand, explicitly extracting such features could get very difficult for less-studied or low-resource languages, and the methods that rely on linguistic features become inherently language dependent. Even a method that considers words as features cannot be completely language independent, since the definition of a word is necessarily language specific. A method that uses only function words as features is also not completely language independent because it needs a list of function words that is specific to a language. When features such as part-of-speech tags are used, as in the work of Jarvis, Bestgen, and Pepper (2013), the method relies on a part-of-speech tagger that 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 p-grams have been used by some of the systems developed for native language identification (Tsur and Rappoport 2007; Tetreault et al. 2012; Jarvis, Bestgen, and Pepper 2013; Popescu and Ionescu 2013). Tsur and Rappoport (2007) have demonstrated that using only the 200 most frequent character bigrams yields a respectable accuracy of 66% on a five-way classification task. The authors explain

their result through the following hypothesis: "the choice of words people make when writing in a second language is strongly influenced by the phonology of their native language." Thus, it seems that character bigrams may capture some phonological preferences. In work where feature ablation results have been reported, the performance with only character p-gram features was modest compared with other types of features (Tetreault et al. 2012). Initially, most work limited the character features to unigrams, bigrams, and trigrams, perhaps because longer p-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 p-grams, up to 9-grams (Jarvis, Bestgen, and Pepper 2013; Popescu and Ionescu 2013). The results presented in this work are also obtained using a range of p-grams. Combining all p-grams in a range 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 more quickly 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 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 (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. Example (1) shows how to compute the p-spectrum kernel between two strings using 2-grams.

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, etc.

Example 1
Given $s =$ "pineapple" and $t =$ "apple pie" over an alphabet Σ, and the substring length $p = 2$, the set of 2-grams that appear in s is denoted by S, and the set of 2-grams that appear in t is denoted by T. The two sets S and T are given by:

$$S = \{\text{“pi”}, \text{“in”}, \text{“ne”}, \text{“ea”}, \text{“ap”}, \text{“pp”}, \text{“pl”}, \text{“le”}\}$$
$$T = \{\text{“ap”}, \text{“pp”}, \text{“pl”}, \text{“le”}, \text{“e\_”}, \text{“\_p”}, \text{“pi””ie”}\}$$

The p-spectrum kernel between s and t can be computed as follows:

$$k_2(s,t) = \sum_{v \in \Sigma^2} \text{num}_v(s) \cdot \text{num}_v(t)$$

where $\Sigma^2 = S \cup T$. If a 2-gram does not appear in one of the strings, the corresponding term in the above sum is zero. As the frequency of each 2-gram in s or t is not greater than one, the p-spectrum kernel is given by $|S \cap T|$, namely, the number of common 2-grams among the two strings. Thus, $k_2(s,t) = 5$.

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, Berg, and Malik 2008; Vedaldi and Zisserman 2010). In the work of Ionescu, Popescu, and Cahill (2014), 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, because 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, because 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 more about the correlation between the p-gram frequencies in the two strings.

Depending on the task, the kernels can yield different results, because one of the kernels can be more suitable than the other for a certain task. For example, the p-spectrum kernel seems to be appropriate for tasks such as authorship identification or text categorization by topic, where the frequency of a p-gram plays an important role. Indeed, the author's style can be easily detected if a certain writing pattern appears several times. On the other hand, a single occurrence of a certain type of mistake could sometimes be enough to make a strong assumption about the native language of a person. From this perspective, the p-gram frequencies could actually contain more noise than useful information. Therefore, using the p-grams presence bits kernel could be more suitable for the NLI task. It is also possible that some mistakes are specific to more than one native language, but the frequency of those mistakes can be different for each L1. In this case, the intersection kernel can capture such language-specific frequency correlations. Various hypotheses in favor of one or the other kernel can be put forward, but instead of choosing a single kernel function, a better approach could be to combine the kernels and use them together. Nevertheless, empirical evidence is necessary for each of these assumptions.

Data normalization helps to improve machine learning performance for various applications. Because the range of values of raw data can have large variation, classifier objective functions will not work properly without normalization. Features are usually normalized through a process called *standardization*, which makes the values of each feature in the data have zero-mean and unit-variance. By normalization, each feature has an approximately equal contribution to the similarity between two samples. Given a kernel k that corresponds to the feature mapping ϕ, the normalized kernel $\hat{k}$ corresponds to the feature map given by:

$$x \longmapsto \phi(x) \longmapsto \frac{\phi(x)}{||\phi(x)||}$$

The kernel normalization can also be done directly on the kernel matrix. To obtain a normalized kernel matrix, each component is divided by the square root of the product of the two corresponding diagonal components:

$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}$$

This is equivalent to normalizing the kernel function as follows:

$$\hat{k}(s_i, s_j) = \frac{k(s_i, s_j)}{\sqrt{k(s_i, s_i) \cdot k(s_j, s_j)}}$$

To ensure a fair comparison of strings of different lengths, normalized versions of the p-spectrum kernel, the p-grams presence bits kernel, and the intersection kernel are used:

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

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

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

Taking into account p-grams of different length and summing up the corresponding kernels, new kernels, termed **blended spectrum kernels**, can be obtained. It is worth mentioning that open source Java implementations of the blended string kernels presented in this work are provided at `http://string-kernels.herokuapp.com`.

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), whereas 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 (LRD; Ionescu 2013), comes from the idea of better adapting rank distance (Dinu and Manea 2006) to string data, in order to capture a better similarity between strings, such as DNA sequences or text. LRD has already shown promising results in computational biology (Ionescu 2013; Dinu, Ionescu, and Tomescu 2014) and native language identification (Popescu and Ionescu 2013; Ionescu 2015).

In order to describe LRD, the following notations are defined. Given a string x over an alphabet Σ, 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 $x[i]x[i+1]\cdots x[j-1]$.

LRD is inspired by rank distance (Dinu and Manea 2006), 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 LRD. Otherwise, the maximal offset m is added to the LRD. 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 this process by exchanging x and y. LRD is formally defined next.

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:

$$\Delta_{left}(x, y) = \sum_{i=1}^{|x|-p+1} \min\{|i - j|$$

such that

$$1 \leq j \leq |y| - p + 1 \text{ and } x[i : i + p] = y[j : j + p]\} \cup \{m\}$$

$$\Delta_{right}(x, y) = \sum_{j=1}^{|y|-p+1} \min\{|j - i|$$

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 size of the p-grams, and to the lengths of the strings. More details about setting up the parameters of LRD are given by Ionescu (2013, 2015). In the experiments, the efficient algorithm of Ionescu (2015) is used to compute LRD.

An upper bound for LRD 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. However, LRD needs to be used as a kernel in the experiments. The classical way to transform a distance or dissimilarity measure into a similarity measure is by using the RBF 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 this 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:

$$g(x) = \sum_{i=1}^{n} \alpha_i \cdot k(x, x_i) \tag{2}$$

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:

$$g(x) = \langle w, x \rangle \tag{3}$$

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:

$$w = \sum_{i=1}^{n} \alpha_i \cdot x_i, \tag{4}$$

given that the kernel function k can be expressed as a scalar product between samples, as follows:

$$k(s, t) = \langle \phi(s), \phi(t) \rangle$$

where ϕ is the embedding feature map associated to the kernel k. An important remark is that not all kernels have a corresponding finite feature space induced by ϕ. It can be trivially observed that the p-spectrum kernel or the p-grams presence bits kernel both have a finite feature space. On the other hand, it is well known (Shawe-Taylor and Cristianini 2004) that the embedding feature map of the RBF kernel induces an infinite feature space. Therefore, the RBF kernel based on LRD can only be used in the dual form. Furthermore, using some kernel functions in the primal form is inherently prohibitive. With enough effort, a finite feature representation for such a kernel can be determined in order to be used in the primal form, but trying to interpret the weights associated with the primal features becomes impossible or at least very difficult. The intersection kernel falls in this category. The language transfer analysis presented in Section 6 is based only on the p-grams presence bits kernel, which is extensively used throughout the experiments, because the weights w corresponding to the primal representation of the p-grams presence bits kernel can be obtained with Equation (4).

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. Because string kernels are based on character p-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. By considering only the 5-grams that occur in a given corpus, there will be fewer features since some p-grams will never appear in the corpus. However, in the experiments conducted on the TOEFL11 corpus, the feature space includes 5-grams along with 6-grams, 7-grams, 8-grams, and 9-grams. The actual number of features generated from this corpus using 5–9 p-grams is precisely $m = 4{,}662{,}520$. As long as the number of samples n is much lower than the number of features m, it can be 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. SVMs 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. KRR 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 the book of Shawe-Taylor and Cristianini (2004). The important fact is that these 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, for instance, kernel discriminant analysis. The KDA classifier is sometimes 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, for instance, when non-native English speakers have acquired, as the second language, a different language from English. Another case of masking is when a native language A is somehow related to two other native languages B and C, in which case the samples that belong to class A can sit in the middle between the samples of classes B and C, as illustrated in Figure 1. In this case, the class in the middle is masked by the other two classes, as it never dominates. KDA can solve such unwanted situations automatically, without having to identify what languages are related by any means, such as geographical position, quantitative linguistic analysis, and so on.

5. Experiments

The main purpose of the experiments is to show that techniques that work at the character level can obtain state-of-the-art performance in NLI and are truly language-independent. The secondary purpose is to determine what kind of kernels and which kernel method works best in the context of NLI.

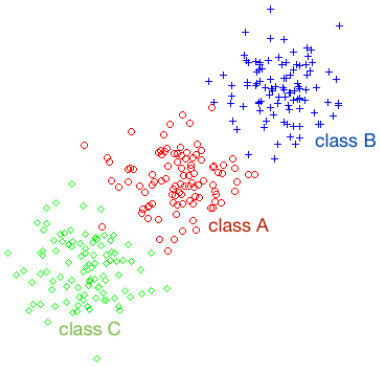

Figure 1
An example with three classes that illustrates the masking problem. Class A is masked by classes B and C.

5.1 Data Set Descriptions

In this article, experiments are carried out on five data sets: a modified version of the ICLEv2 corpus (Granger, Dagneaux, and Meunier 2009); the ETS Corpus of Non-Native Written English, or TOEFL11 (Blanchard et al. 2013); the TOEFL11-Big corpus as used by Tetreault et al. (2012); a subset of the second version of the Arabic Learner Corpus (ALC) (Alfaifi, Atwell, and Hedaya 2014); and a subset of the ASK Corpus (Tenfjord, Meurer, and Hofland 2006). A summary of the corpora is given in Table 1.

The ICLEv2 is a corpus of essays written by highly proficient non-native college-level students of English. 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 in the experiments. This version of the corpus contains 110 essays each for seven 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, Blanchard, and Cahill 2013). The TOEFL11 corpus contains a balanced distribution of essays per prompt (topic) per native language. 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.

Table 1
Summary of corpora used in the experiments.

Corpus	L2	L1 Languages	Documents
ICLEv2	English	7	770
TOEFL11	English	11	12,100
TOEFL11-Big	English	11	87,502
ALC Subset	Arabic	7	329
ASK Corpus Subset	Norwegian	7	1,400

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 87,000 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.

The second version of the Arabic Learner Corpus was recently introduced by Alfaifi, Atwell, and Hedaya (2014). The corpus includes essays by Arabic learners studying in Saudi Arabia. There are 66 different mother tongue (L1) representations in the corpus, but the majority of these have fewer than 10 essays. To compare the string kernels approach with the state-of-the-art method (Malmasi and Dras 2014a), the subset of ALC used by Malmasi and Dras (2014a) is also used in this article. The subset is obtained by considering only the top seven native languages by number of essays. The ALC subset used in the experiments contains 329 essays in total, which are not evenly distributed per native language, as shown in Table 2.

The ASK Corpus is a language learner corpus of Norwegian as a second language developed by Tenfjord, Meurer, and Hofland (2006). It contains 1,938 essays collected from language tests on two different proficiency levels. There are 9 native languages in the corpus, but only 7 of them are equally represented by 200 essays each. To avoid any bias from the uneven distribution of documents per language, the experiments presented in this work include only the 7 languages that contain 200 essays each: English, Dutch, German, Polish, Russian, Serbo-Croatian, and Spanish. The corpus has been tokenized, and no raw version is available.

5.2 Parameter Tuning and Implementation Choices

In the string kernels approach proposed in this work, documents or essays from the corpora are treated as strings. Therefore, the notions of *string* and *document* are used interchangeably. 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 whitespace characters (space, tab, new line, and so on) with a single space character. This normalization is needed in order to prevent an incorrect measure of similarity between texts, simply as a result of different spacing. All uppercase letters were converted to the corresponding lowercase ones. The letter case normalization has no effect on the documents of the ALC subset, because there are no distinct upper and lower case letter forms in the Arabic alphabet.

Table 2
Distribution of the documents per native language in the ALC subset.

Language	Documents
Chinese	76
English	35
French	44
Fulani	36
Malay	46
Urdu	64
Yoruba	28
Total	329

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 p-grams that gives the most accurate results. Each language was separately considered to determine the optimal range. This is motivated by how different the Arabic writing system is from English and Norwegian. For example, short vowels can be omitted from the Arabic writing, and consequently produce shorter words. There could also be differences between English and Norwegian, because Norwegian has a productive compound noun system that may occasionally result in very long words, although of course the two languages are more closely related to each other than to Arabic, because they belong to the branch of Germanic languages.

First, a 10-fold cross-validation procedure was carried out on the TOEFL11 training set. All the p-grams in the range 2–10 were evaluated. Furthermore, experiments with different blended kernels were conducted to see whether combining p-grams of different lengths could improve the accuracy. The best results were obtained when all the p-grams with length in the range 5–8 were used. Other authors (Bykh and Meurers 2012; Popescu and Ionescu 2013) also report better results by using p-grams with the length in a range, rather than using p-grams of fixed length. Consequently, the results reported on the English corpora are based on blended string kernels based on 5–8 p-grams. Using a similar procedure on the ASK corpus, the range of p-grams found to work best was again 5–8. A 10-fold cross-validation procedure was also conducted on the ALC subset of 329 documents in order to evaluate the p-grams in the range 2–7. The best performance was obtained using 4-grams. In general, a blended spectrum of p-grams produces better results, and therefore, the combination of 3–5 p-grams (centered around 4-grams) was chosen for the rest of the experiments on the ALC corpus. Preliminary results on the Arabic corpus indicate that using the range of 3–5 p-grams gives a roughly 2% improvement over using p-grams of length 4 alone. Other ranges of p-grams were not evaluated to prevent any overfitting on the ALC corpus. An interesting note is that the only difference between the systems evaluated on the English corpora and those evaluated on the Arabic corpus is that they are based on different p-gram lengths. On the other hand, the range of p-grams is identical for the English and the Norwegian corpora, probably because the two languages are more closely related to each other than to Arabic. Table 3 shows the average word length for the three corpora on which the range of p-grams was independently tuned. Remarkably, the average word lengths computed in the current study are consistent with other statistics reported in literature. Indeed, according to a study conducted by Intellaren,[2] the average word length in the Quran (Arabic) is 4.25 letters, and, according to Manning, Raghavan, and Schütze (2008), the average English word length is about 4.5 letters. In general, it seems that the range of p-grams is related to the average word length in a specific language. More precisely, the p-grams having the same length as the average word or the ones that are slightly

2 http://www.intellaren.com/articles/en/qss.

Table 3
Average word length and optimal p-gram range for the TOEFL11 corpus (English L2), the ALC subset (Arabic L2), and the ASK corpus (Norwegian L2). Spaces and punctuation were not taken into consideration when the average word lengths were computed.

Corpus	Average word length	Range of p-grams
TOEFL11	4.46	5–8
ALC Subset	4.36	3–5
ASK Corpus Subset	4.34	5–8

longer produce better accuracy rates. This is an encouraging result because it confirms that language-specific knowledge (e.g., average word length) is not required (although it can be helpful), but can be empirically established as described above, if necessary.

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. Because 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, the ICLE corpus, the ALC subset, and the ASK corpus.

5.3 Experiment on TOEFL11 Corpus

This section describes the results on the TOEFL11 corpus and, for the sake of completion, it also includes results for the 2013 *Closed* NLI Shared Task. 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 approach presented here.

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 4.

The results presented in Table 4 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 provides significantly lower accuracy rates, but it is able to improve the performance when it is 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 percentage points better than the state-of-the-art system of Jarvis, Bestgen, and Pepper (2013) based on SVM and word features, this being the top scoring system in the 2013 NLI Shared Task. It is also 2.6 percentage points 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 an accuracy 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 percentage points above the state-of-the-art system of Tetreault et al. (2012) and 0.4 percentage

Table 4
Accuracy rates on TOEFL11 corpus (English L2) 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.

Method	Dev	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, Bestgen, and Pepper 2013)	-	**84.5**%	83.6%
Ensemble model and CFG (Bykh and Meurers 2014)	-	-	84.8%
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%

points below the top scoring system of Jarvis, Bestgen, and Pepper (2013). The empirical results obtained in this experiment demonstrate that the approach proposed in this article can reach state-of-the-art accuracy levels.

The results presented in Table 4 are obtained on the data released for the 2013 NLI Shared Task. The documents provided for the task were already tokenized. Access to the raw text documents was not provided during the task. As the TOEFL11 corpus was later released through LDC,[3] the raw text documents also became available for use. It is particularly interesting to evaluate the various string kernel systems on the raw text documents, in order to determine whether the string kernels approach is indeed neutral to any linguistic theory. The process of breaking a stream of text up into tokens is not an easy task for some languages and it is often based on linguistic theories. Even if tokenization is fairly straightforward in languages that use spaces to separate words, there are many edge cases including contractions and hyphenated words that require linguistic knowledge. Therefore, analyzing the results of the string kernels approach when tokenization is not being used at all will reveal if the linguistic knowledge helps to improve the performance or not. Table 5 reports the results on the raw text documents of various strings kernels combined either with KRR and KDA. An important remark is that the tokenization process can influence the average word length in the corpus. For instance, "don't" is divided into two tokens, namely, "do" and "n't".[4] Therefore, the range of p-grams used for computing the blended string kernels was adjusted based on the development set for the raw text documents. All p-grams in the range 3–9 were evaluated and the best results were obtained when all the p-grams with length in the range 5–9 were used. Not surprisingly, the range of p-grams for the raw text documents is fairly close to the range of p-grams that worked best for the tokenized text.

The empirical results shown in Table 5 indicate that the presence bits kernel and the kernel based on LRD obtain better results on the raw text documents. On the other hand the results of the intersection kernel are roughly the same, or only slightly better than, the results presented in Table 4. When the various kernels are mixed together, the results are nearly identical to the results obtained on tokenized text documents. Overall, the results are better when the string kernels are computed on the raw text, although the improvements are not considerably different, since the accuracy difference is less than 1 percentage point in all cases. However, the improvement could be the result of removing tokenization or the results of using a different range of p-grams. To find which of these two hypotheses is responsible for the performance gain, the presence bits kernel was computed on the raw text documents using a range of 5–8 p-grams. The accuracy obtained by KRR and $\hat{k}^{0/1}_{5-8}$ on raw text was 82.6%, which is still better than the accuracy of the very same system when tokenization was included (82.0%). Therefore, we can conclude that both hypotheses are responsible for the improved results. According to some linguistic theories, it is recommended to separate the text into tokens before any other processing is done. However, this is not necessary in the case of string kernels. Indeed, the empirical results confirmed that the approach based on character p-grams does not rely on any linguistic theory to reach state-of-the-art performance levels.

It is worth mentioning that a significance test performed by the organizers of the 2013 NLI Shared Task showed that the top systems that participated in the competition are not essentially different. Their conclusion also implies that adding or removing text tokenization in the string kernel approach does not have a significant influence

3 https://catalog.ldc.upenn.edu/LDC2014T06.

4 Note that in some tokenization systems this would even be three tokens: "don", "'", and "t".

Table 5
Accuracy rates on the raw text documents of the TOEFL11 corpus (English L2) of various classification systems based on string kernels. 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.

Method	Dev	10-fold CV	Test
KRR and $\hat{k}^{0/1}_{5-9}$	86.3%	82.6%	82.8%
KRR and $\hat{k}^{\cap}_{5-9}$	84.5%	82.5%	82.6%
KRR and $\hat{k}^{LRD}_{5-9}$	79.9%	77.9%	77.6%
KRR and $\hat{k}^{0/1}_{5-9} + \hat{k}^{LRD}_{5-9}$	86.1%	82.6%	83.1%
KRR and $\hat{k}^{\cap}_{5-9} + \hat{k}^{LRD}_{5-9}$	84.9%	82.4%	82.1%
KRR and $\hat{k}^{0/1}_{5-9} + \hat{k}^{\cap}_{5-9}$	85.6%	82.9%	82.6%
KRR and $a_1\hat{k}^{0/1}_{5-9} + a_2\hat{k}^{\cap}_{5-9}$	85.6%	82.9%	82.6%
KDA and $\hat{k}^{0/1}_{5-9}$	86.2%	83.7%	84.6%
KDA and $\hat{k}^{\cap}_{5-9}$	85.2%	83.6%	84.3%
KDA and $\hat{k}^{LRD}_{5-9}$	81.2%	79.1%	79.6%
KDA and $\hat{k}^{0/1}_{5-9} + \hat{k}^{LRD}_{5-9}$	86.6%	83.9%	84.7%
KDA and $\hat{k}^{\cap}_{5-9} + \hat{k}^{LRD}_{5-9}$	85.9%	83.5%	83.9%
KDA and $\hat{k}^{0/1}_{5-9} + \hat{k}^{\cap}_{5-9}$	85.9%	**84.1**%	85.1%
KDA and $a_1\hat{k}^{0/1}_{5-9} + a_2\hat{k}^{\cap}_{5-9}$	86.1%	**84.1**%	**85.3**%
KDA and $\hat{k}^{0/1}_{5-9} + \hat{k}^{\cap}_{5-9} + \hat{k}^{LRD}_{5-9}$	**86.7**%	84.0%	84.9%

on the results, even if they appear to be slightly better when tokenization is not used on the TOEFL11 corpus. Nevertheless, the rest of the experiments on English corpora are conducted on tokenized text documents, to ensure a fair comparison with the other state-of-the-art approaches that use text tokenization as a preprocessing step. The second reason for applying the string kernel approach on tokenized texts is that the range of p-grams is smaller (5–8 instead of 5–9), which translates into a faster technique that requires less time to compute the blended spectrum kernels. Note, though, that because the texts in the ALC corpus are not tokenized, the string kernel approach is applied directly to the raw text documents. In other words, the string kernel approach is evaluated on tokenized text documents as long as these documents are included in the corpus. On the other hand, the language transfer analysis given in Section 6 includes only the features computed on the raw text documents of the TOEFL11 corpus, in order to avoid any confusion between the language transfer patterns and the patterns induced by tokenization. For instance, adding spaces before punctuation is part of the tokenization process, but it could be mistaken for a language transfer pattern as well.

5.4 Experiment on ICLE Corpus

The results on the ICLEv2 corpus using a 5-fold cross-validation procedure are summarized in Table 6. 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 6 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

Table 6
Accuracy rates on ICLEv2 corpus (English L2) of various classification systems based on string kernels compared with a state-of-the-art approach. The accuracy rates are reported using a 5-fold cross-validation (CV) procedure. The best accuracy rate is highlighted in **bold**.

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%

of Tetreault et al. (2012). Although 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 the kernel based on LRD. This represents a 1.2 percentage point 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 ICLEv2 corpus show that the string kernels approach can reach state-of-the-art accuracy levels, although a paired-sample Student's t-test indicated that the results are not significantly different. 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 p-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 Experiment on TOEFL11-Big Corpus

The presence bits kernel and the intersection kernel are evaluated on the TOEFL11-Big corpus using the 10-fold CV procedure. The folds are chosen according to Tetreault et al.

(2012), in order to fairly compare the string kernels with their state-of-the-art ensemble model. The kernel based on LRD is not evaluated in this experiment on purpose, because it obtained lower accuracy rates in the TOEFL11 and ICLE experiments. The second reason for not including LRD is that it is more computationally expensive than the presence bits kernel or the intersection kernel.

As in the previous experiments, both KRR and KDA were tested on TOEFL11-Big, but KDA was not trained using all the training samples because it runs out of memory. The machine used for this experiment has 256 GB of RAM, and it seems that training KDA on 78,000 samples requires slightly more memory. However, a KDA classifier can successfully be trained on 75,000 samples on the same machine. Therefore, the results of KDA presented in this section are obtained by training it on a random sample of 75,000 samples selected during each fold. The results of KDA might be slightly different (but probably less than 1 percentage point) if the entire available training data would be used. The regularization parameters of KRR and KDA were tuned such that an accuracy of roughly 98% is obtained on the training set. In other words, the classifiers are allowed to disregard around 2% of the training documents to prevent overfitting. From this perspective, the disregarded training documents are not considered helpful for the NLI task. Preliminary results showed that trying to fit more of the training data leads to lower accuracy rates (not more than 1–2 percentage points, though) in the CV procedure.

The results obtained by the presence bits kernel and by the intersection kernel are presented in Table 7. The two kernels are also combined together. When considering the individual kernels, KDA seems to work much better than KRR, even if it does not use the entire available training data set. For instance, the KDA classifier based on the intersection kernel attains the same accuracy as the state-of-the-art ensemble model of Tetreault et al. (2012), whereas KRR based on the same kernel reaches an accuracy of only 82.3%, which is 2.3 percentage points lower than the state-of-the-art (84.6%). On the other hand, KRR overturns the results when the two kernels are combined through MKL. Indeed, KRR based on the sum of $\hat{k}^{0/1}_{5-8}$ and $\hat{k}^{\cap}_{5-8}$ reaches an accuracy of 84.8%, and KDA based on the kernel combination obtains 84.7%. Both of them are better than the state-of-the-art model. Although an improvement of 0.2 percentage point or 0.1 percentage point over the state-of-the-art does not look like much, the number

Table 7
Accuracy rates on TOEFL11-Big (English L2) corpus of various classification systems based on string kernels compared with a state-of-the-art approach. The accuracy rates are reported using a 10-fold cross-validation (CV) procedure. The best accuracy rate is highlighted in **bold**.

Method	**Test**
Ensemble model (Tetreault et al. 2012)	84.6%
KRR and $\hat{k}^{0/1}_{5-8}$	82.6%
KRR and $\hat{k}^{\cap}_{5-8}$	82.3%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	**84.8**%
KDA and $\hat{k}^{0/1}_{5-8}$	84.5%
KDA and $\hat{k}^{\cap}_{5-8}$	84.6%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	84.7%

of samples in the TOEFL11-Big data set can tell a different story, since 0.1% means roughly 90 samples. From this point of view, KRR based on the kernel combination assigns the correct labels for more than 170 extra samples compared to the state-of-the-art system. Moreover, these improvements are obtained without using any feature engineering, which represents an advantage over the standard NLP approaches. To draw a conclusion on the empirical results, the idea of combining kernels through MKL proves to be extremely helpful for boosting the performance of the classification system. KDA seems to be more stable with respect to the type of kernel, whereas KRR gives the top accuracy only when the kernels are combined.

5.6 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 state-of-the-art ensemble model (Tetreault et al. 2012) was trained and tested in exactly the same manner. The goal of this cross-corpus evaluation is to show that the string kernel approach does not perform well simply because of potential topic bias in the corpora. Again, the kernel based on LRD was not included in this experiment because 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 8. The same regularization parameters determined to work well on the TOEFL11 development set were used.

The most interesting fact is that all the proposed systems are at least 30 percentage points better than the state-of-the-art system. Considering that the TOEFL11-Big corpus contains 87,000 samples, the 30 percentage point improvement is significant without any doubt. Diving into details, we can see that the results obtained by KRR are higher than those obtained by KDA. However, both methods perform very well compared with the state-of-the-art. Again, kernel combinations are better than each of their individual kernels alone.

Table 8
Accuracy rates on TOEFL11-Big corpus (English L2) 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.

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%

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. The topics (prompts) from TOEFL11 are different from the topics from TOEFL11-Big, and it becomes clear that a method that uses words as features is strongly affected by topic variations, because the distribution of words per topic can be completely different. But mistakes that reveal the native language can be captured by character p-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 percentage points 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.

5.7 Experiment on ALC Subset Corpus

The string kernels approach is evaluated on Arabic data in order to demonstrate that the approach is language independent by showing that it can obtain state-of-the-art results. Malmasi and Dras (2014a) are the first (and only) to show NLI results on Arabic data. In their evaluation, the 10-fold CV procedure was used to evaluate an SVM model based on several types of features including CFG rules, function words, and part-of-speech p-grams. To directly compare the string kernel systems with the approach of Malmasi and Dras (2014a), the same folds should have been used. Because the folds are not publicly available, two alternative solutions are adopted to fairly compare the two NLI approaches. First of all, each classification system based on string kernels is evaluated by repeating the 10-fold CV procedure for 20 times and averaging the resulted accuracy rates. The folds are randomly selected at each trial. This helps to reduce the amount of accuracy variation introduced by using a different partition of the data set for the 10-fold CV procedure. To provide an idea of the amount of variation in each trial, the standard deviations for the computed average accuracy rates are also reported. Second of all, the leave-one-out (LOO) cross-validation procedure is also adopted because it involves a predefined partitioning of the data set. Furthermore, the LOO procedure can easily be performed on a small data set, such as the subset of 329 samples of the ALC. Thus, the LOO procedure is more suitable for this NLI experiment, because it is straightforward to compare newly developed systems with the previous state-of-the-art systems. Malmasi and Dras (2014a) were kindly asked to re-evaluate their system using the LOO CV procedure and they readily provided additional results, included in Table 9. Lastly, a studentized bootstrap procedure based on 200 iterations was employed to provide confidence intervals with a confidence level of 95%. The bootstrap procedure is suitable for small data sets such as the ALC subset.

Table 9 presents the results of the classification systems based on string kernels in contrast to the results of the SVM model based on several combined features. The results clearly indicate the advantage of using an approach that works at the character level. When the 10-fold CV procedure is used, the average accuracy rates of the systems based on string kernels range between 45.9% and 56.8%. The lowest scores are obtained by LRD, whereas the presence bits kernel and the intersection kernel obtain better results. Even so, the kernel based on LRD is far better than the model of Malmasi and Dras (2014a). Combining the kernels through MKL again proves to be a good idea. Certainly, the top scoring system in the 10-fold CV experiment is the KRR based on

Table 9
Accuracy rates on ALC subset (Arabic L2) of various classification systems based on string kernels compared with a state-of-the-art approach. The accuracy rates are reported using a studentized bootstrap procedure and two cross-validation (CV) procedures, one based on 10 folds and one based on leave-one-out (LOO). The 10-fold CV procedure was repeated for 20 times and the results were averaged to reduce the accuracy variation introduced by randomly selecting the folds. The standard deviations for the computed average accuracy rates are also given. The bootstrap procedure is based on 200 iterations, and the reported confidence intervals are based on a 95% confidence level. The best accuracy rate is highlighted in **bold**.

Method	10-fold CV	LOO CV	Bootstrap
SVM and combined features (Malmasi and Dras 2014a)	41.0%	41.6%	-
KRR and $\hat{k}^{0/1}_{3-5}$	55.5% ± 1.3	58.7%	48.83–49.91%
KRR and $\hat{k}^{\cap}_{3-5}$	55.2% ± 1.4	56.8%	48.56–49.29%
KRR and $\hat{k}^{LRD}_{3-5}$	50.1% ± 1.2	51.4%	47.67–48.75%
KRR and $\hat{k}^{0/1}_{3-5} + \hat{k}^{LRD}_{3-5}$	54.9% ± 1.2	56.8%	48.86–49.97%
KRR and $\hat{k}^{\cap}_{3-5} + \hat{k}^{LRD}_{3-5}$	54.7% ± 1.4	57.2%	48.44–49.86%
KRR and $\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5}$	**56.8% ± 1.3**	**59.3%**	**48.93–50.14%**
KRR and $\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5} + \hat{k}^{LRD}_{3-5}$	56.5% ± 1.6	59.0%	48.60–49.89%
KDA and $\hat{k}^{0/1}_{3-5}$	55.3% ± 1.1	57.4%	47.34–48.57%
KDA and $\hat{k}^{\cap}_{3-5}$	51.2% ± 1.5	53.2%	46.02–47.20%
KDA and $\hat{k}^{LRD}_{3-5}$	45.9% ± 1.1	47.1%	42.59–43.58%
KDA and $\hat{k}^{0/1}_{3-5} + \hat{k}^{LRD}_{3-5}$	52.7% ± 1.3	54.4%	46.00–47.15%
KDA and $\hat{k}^{\cap}_{3-5} + \hat{k}^{LRD}_{3-5}$	50.2% ± 1.3	51.1%	45.66–47.07%
KDA and $\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5}$	53.4% ± 1.2	55.3%	46.55–47.93%
KDA and $\hat{k}^{0/1}_{3-5} + \hat{k}^{\cap}_{3-5} + \hat{k}^{LRD}_{3-5}$	53.8% ± 1.1	54.7%	46.01–47.26%

the sum of the presence bits kernel and the intersection kernel. Its accuracy (56.8%) is 15.8 percentage points above the accuracy of the SVM based on combined features (41.0%). An important remark is that the standard deviations computed over the 20 trials are between 1.1% and 1.6% for all systems, which means the amount of accuracy variation is too small to have an influence on the overall conclusion. Furthermore, all the results obtained using the 10-fold CV procedure are consistent with the results obtained using the leave-one-out CV procedure or the bootstrap procedure. The accuracy rates are generally lower when bootstrap is used for evaluation, because the bootstrap procedure is a rather more pessimistic way of estimating the performance compared to the CV procedures. With no exception, the models reach better accuracy rates when the LOO procedure is used, most likely because there are more samples available for training. When the LOO CV procedure is used, the systems based on string kernels attain accuracy rates that range between 47.1% and 59.3%. Better accuracy rates are obtained when the kernels are combined. When $\hat{k}^{0/1}_{3-5}$ and $\hat{k}^{\cap}_{3-5}$ are summed together and the training is performed by KRR, the best accuracy rate (59.3%) is obtained, which brings an improvement of 17.7 percentage points over accuracy of the SVM model (41.6%). Regarding the classification systems, KRR obtains better results than KDA for every kernel type, even if the differences are not that high. Still, both KRR and KDA give results that are much better than the SVM of Malmasi and Dras (2014a). In conclusion,

each and every classification systems based on string kernels is significantly better that the SVM model of Malmasi and Dras (2014a) on the Arabic data set.

The topic distribution per native language is unknown for the ALC subset, so Malmasi and Dras (2014a) removed any lexical information from their model to prevent any topic bias. However, Alfaifi, Atwell, and Hedaya (2014) state in their paper presenting the second version of the Arabic Learner Corpus that: "We decided to choose two common genres in the learner corpora, narrative and discussion. For the written part, learners had to write narrative or discussion essays under two specific titles which were likely to suit both native and non-native learners of Arabic, entitled *a vacation trip* for the narrative and *my study interest* for the discussion type," which means that, at a coarse-grained level, there are two main topics (prompts) in the corpus. However, the topic distribution can also be discussed at a fine-grained level. Naturally, each of the two main topics can be further divided into a broad range of subtopics. Because there is a large number of potential subtopics, it is unlikely that their distribution per native language is biased in such a way that it will have an impact on the results. Considering the coarse-grained topic distribution, there is no chance that the accuracy rates obtained by the various systems based on string kernels are due any kind of topic bias, since there are only two topics and seven native languages in the ALC subset used in this experiment. Even in the worst case scenario, if the documents for each L1 were to belong to a single topic, a method that relies on topic distribution to predict the native language could obtain at most 42.6%, assuming the most frequent L1 languages in the corpus (Chinese and Urdu) belong to different topics. More precisely, 42.6% can be obtained only when the texts written by Chinese speakers belong to one topic, the texts written by Urdu speakers to the other topic, and the classifier based on topic distribution can predict the right topic with 100% accuracy for each of the two native languages. It becomes obvious that the best result reported in this section using string kernels (59.3%) cannot be obtained with a classifier that relies on the topics distribution per native language. This observation is consistent with the results obtained in the cross-corpus experiment (Section 5.6), which demonstrate that the string kernel approach naturally avoids topic bias. In light of these comments, it is worth mentioning that the SVM model of Malmasi and Dras (2014a) could benefit from lexical information without the risk of introducing topic bias. Indeed, their model extended to include character 3-grams reaches an accuracy of 51.2% using the LOO procedure. Still, this results is not as good as using character p-grams alone, as the results of string kernels presented in Table 9 indicate.

5.8 Experiment on ASK Corpus

Pepper (2012) conducted an extensive set of experiments on the ASK corpus. The 10-fold CV procedure and the leave-one-out CV procedure were alternatively used to evaluate various LDA models based on linguistic features. Pepper used only 100 essays per native language and formed several subsets each of five languages. All the subsets included four languages in common, namely, English, German, Polish, and Russian. The fifth language was different in each subset. The three subsets that included Spanish, Dutch, and Serbo-Croatian as the fifth language have also been used in the present work in order to compare the results of string kernels with the LDA model of Pepper (2012). For a direct comparison, the same 100 essays per native language should have been used, but this information is not available. For this reason, the string kernels are evaluated by repeating the LOO CV procedure 20 times, and in each trial 100 essays per native language are randomly selected from the 200 essays available for each language. Table 10 reports the average accuracy rates along with their standard deviations which

give some indication about the amount of variation in each trial. The results obtained on the subset that includes Spanish along with English, German, Polish, and Russian are given in the second column. In a similar way, the results obtained on the subset that includes Dutch as the fifth language are listed in the third column, while the results obtained on the subset that includes Serbo-Croatian as the fifth language are given in the last column. The best results obtained by Pepper (2012) using the LOO CV procedure are listed in the first row. An important remark is that Pepper (2012) reported much lower accuracy rates when the 10-fold CV procedure was used. Because of this, the comparison is carried out only by using the LOO CV procedure, which also helps to reduce the amount of variation in the 20 trials. However, it must be mentioned the results of string kernels are only 2 percentage points lower when the 10-fold CV procedure is used, but any further details of that experiment are not reported here.

The empirical results presented in Table 10 are consistent with the results on the ALC corpus. Indeed, all the systems based on string kernels are significantly better than the state-of-the-art approach, in all three experiments. KDA and KRR seem to produce fairly similar results, but there is an interesting difference between the two classifiers that stands out. This difference refers to the fact that KRR produces better results when the sum of $\hat{k}^{0/1}_{5-8}$ and $\hat{k}^{\cap}_{5-8}$ is used, whereas KDA produces better results when the kernel

Table 10
Accuracy rates on three subsets of five languages of the ASK corpus (Norwegian L2) of various classification systems based on string kernels compared with a state-of-the-art approach. All subsets include samples of English, German, Polish, and Russian (EGPR). The fifth language is different in each subset. The first subset includes Spanish (SP), the second one includes Dutch (DU), and the last subset includes Serbo-Croatian (SC) as the fifth language. The accuracy rates for each subset are reported using the leave-one-out cross-validation procedure, which was repeated 20 times and the results were averaged to reduce the accuracy variation introduced by randomly selecting the documents (100 per language). The standard deviations for the computed average accuracy rates are also given. The best accuracy rate for each subset of five languages is highlighted in **bold**.

Method	EGPR+SP	EGPR+DU	EGPR+SC
LDA (Pepper 2012)	51.2%	55.0%	55.0%
KRR and $\hat{k}^{0/1}_{5-8}$	$67.4\% \pm 2.1$	$68.2\% \pm 1.8$	$68.1\% \pm 2.0$
KRR and $\hat{k}^{\cap}_{5-8}$	$67.5\% \pm 1.7$	$68.6\% \pm 2.0$	$68.2\% \pm 2.0$
KRR and $\hat{k}^{LRD}_{5-8}$	$62.2\% \pm 1.9$	$62.9\% \pm 1.9$	$61.5\% \pm 2.1$
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	$66.6\% \pm 2.1$	$67.5\% \pm 1.6$	$65.6\% \pm 1.4$
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	$66.2\% \pm 2.1$	$67.7\% \pm 1.9$	$65.7\% \pm 2.0$
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	$\mathbf{67.8\% \pm 1.9}$	$69.0\% \pm 1.4$	$\mathbf{68.5\% \pm 1.8}$
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	$67.5\% \pm 1.9$	$68.4\% \pm 2.0$	$67.1\% \pm 1.9$
KDA and $\hat{k}^{0/1}_{5-8}$	$67.2\% \pm 1.7$	$68.7\% \pm 1.4$	$67.9\% \pm 1.5$
KDA and $\hat{k}^{\cap}_{5-8}$	$67.3\% \pm 2.0$	$68.8\% \pm 1.4$	$67.3\% \pm 2.2$
KDA and $\hat{k}^{LRD}_{5-8}$	$61.9\% \pm 1.8$	$63.4\% \pm 1.7$	$61.0\% \pm 2.0$
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	$66.6\% \pm 2.1$	$67.4\% \pm 1.5$	$65.7\% \pm 1.4$
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	$65.9\% \pm 1.8$	$67.2\% \pm 1.2$	$65.9\% \pm 1.8$
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	$67.4\% \pm 1.8$	$68.5\% \pm 1.9$	$67.2\% \pm 1.6$
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	$67.7\% \pm 1.4$	$\mathbf{69.1\% \pm 1.9}$	$68.0\% \pm 1.3$

based on LRD is also added into the sum. When the kernels are used independently, it can be observed that LRD produces accuracy rates that are nearly 5 percentage points below the accuracy rates of the intersection kernel and the presence bits kernel, respectively. Nevertheless, the kernel based on LRD is still significantly better than the LDA model used by Pepper (2012). For instance, when Spanish is used as the fifth language, all the string kernels, including the one based on LRD, attain accuracy rates that are at least 10 percentage points better than LDA. In all three evaluation sets, the best results are obtained when kernels are combined together. The best accuracy on the first subset (67.8%) is given by the KRR based on the kernel combination of $\hat{k}^{0/1}_{5-8}$ and $\hat{k}^{\cap}_{5-8}$. Compared with the LDA model, the accuracy improvement is 16.6 percentage points. Likewise, the best accuracy on the third subset (68.5%) is again obtained by the KRR based on the kernel combination of $\hat{k}^{0/1}_{5-8}$ and $\hat{k}^{\cap}_{5-8}$. This time, the accuracy improvement over the state-of-the-art LDA model is 13.5 percentage points. The best accuracy on the second subset of five languages (69.1%) is obtained by the KDA based on the combination of all three string kernels. The accuracy improvement over the LDA model is 14.1 percentage points, which is consistent with the improvement demonstrated on the other two subsets that include Spanish and Serbo-Croatian, respectively. Even if the string kernels are significantly better than the LDA model, the two distinct approaches seem to agree on the difficulty of each subset. Indeed, both approaches produce better results when Dutch is used as the fifth language and worse results when Spanish is the fifth language. This could indicate that Dutch native speakers writing in Norwegian can be more easily distinguished than Spanish speakers writing in Norwegian. Although one might expect Dutch, German, and English speakers to be more similar (and therefore potentially more difficult to distinguish) than, say, English or Spanish speakers writing in Norwegian, this does not appear to be the case, given that the overall performance is better when Dutch is added to the mix of English, German, Polish, and Russian. What can be said for sure is that the approach based on string kernels attains a significant performance improvement over the LDA based on linguistic features. The standard deviations computed over the 20 trials are less than 2.2% for all systems, which indicates that the amount of accuracy variation is small enough to support the conclusion that the string kernel approach works better.

The string kernels are further evaluated on all seven languages that are equally represented in the ASK corpus. The purpose of this evaluation is to provide an easy way of comparing other newly developed systems (in the future) with the systems based on string kernels. Two standard cross-validation procedures are used. First, each classification system based on string kernels is evaluated by repeating the 10-fold CV procedure 20 times and averaging the resulted accuracy rates. The folds are randomly selected at each trial. Second, the leave-one-out CV procedure was also adopted because it involves a predefined partitioning of the data set. The results for the seven-way classification task are given in Table 11.

The results presented in Table 11 are similar to those presented in Table 10. The extra number of documents per native language (200 instead of 100) seems to compensate for having to discriminate between seven languages instead of five. When the 10-fold CV procedure is used, the average accuracy rates of the systems based on string kernels range between 60.5% and 68.2%. The lowest scores are obtained by LRD, while the presence bits kernel and the intersection kernel obtain better results. Combining the kernels through MKL proves to work again as a robust approach that usually improves performance. It must be noted that combining the LRD kernel with either one of the other kernels ($\hat{k}^{0/1}_{5-8}$ or $\hat{k}^{\cap}_{5-8}$) provides slightly lower results than using the respective

Table 11
Accuracy rates on the ASK corpus (Norwegian L2) of various classification systems based on string kernels. The accuracy rates are reported using two cross-validation (CV) procedures, one based on 10 folds and one based on leave-one-out (LOO). The 10-fold CV procedure was repeated 20 times and the results were averaged to reduce the accuracy variation introduced by randomly selecting the folds. The standard deviations for the computed average accuracy rates are also given. The best accuracy rate is highlighted in **bold**.

Method	10-fold CV	LOO CV
KRR and $\hat{k}^{0/1}_{5-8}$	67.1% ± 0.6	68.2%
KRR and $\hat{k}^{\cap}_{5-8}$	67.4% ± 0.6	68.1%
KRR and $\hat{k}^{LRD}_{5-8}$	60.5% ± 0.7	62.0%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	66.7% ± 0.5	67.5%
KRR and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	66.9% ± 0.6	68.7%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	67.6% ± 0.5	68.1%
KRR and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	67.9% ± 0.6	69.0%
KDA and $\hat{k}^{0/1}_{5-8}$	67.6% ± 0.7	69.2%
KDA and $\hat{k}^{\cap}_{5-8}$	67.7% ± 0.7	69.1%
KDA and $\hat{k}^{LRD}_{5-8}$	60.8% ± 0.5	61.9%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{LRD}_{5-8}$	66.7% ± 0.8	68.0%
KDA and $\hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	67.0% ± 0.5	68.3%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8}$	**68.2**% ± **0.7**	69.2%
KDA and $\hat{k}^{0/1}_{5-8} + \hat{k}^{\cap}_{5-8} + \hat{k}^{LRD}_{5-8}$	67.9% ± 0.5	**69.6**%

kernels independently. On the other hand, the top scoring system in the 10-fold CV experiment is the KDA based on the sum of the presence bits kernel and the intersection kernel. All the results obtained using the LOO CV procedure are consistently higher than the results obtained using the 10-fold CV procedure, because there are more samples available for training. When the LOO CV procedure is used, the systems based on string kernels reach accuracy rates that range between 61.9% and 69.6%. The best system proves to be the KDA based on the sum of the presence bits kernel, the intersection kernel, and the kernel based on LRD. Overall, KDA seems to produce better results than KRR in the seven-way classification task, but none of the two classifiers seems to be better than the other in the five-way classification tasks. Nevertheless, all the results produced by string kernels are very good, especially when compared with the LDA based on linguistic features (Pepper 2012). The results on the Norwegian corpus along with the results on the Arabic corpus demonstrate that the string kernel approach is language independent.

5.9 Discussion

The state-of-the-art accuracy rates have been surpassed in all the experiments presented in this work, sometimes by a very large margin. The idea of combining the string kernels, either by kernel sum or by kernel alignment, proved to be extremely useful, always producing the best NLI performance. Certainly, the best system presented in this work is based on combining the intersection and the presence string kernels

through MKL and on deciding the class label either with KDA or KRR. Furthermore, the experiments on the five corpora presented herein show that the approach based on string kernels generalizes well across corpora, topics, and languages. However, it should also be pointed out that the corpora available for these kinds of studies are typically academic in nature (i.e., from language learners). Applying these models to other genres of text (tweets, for example) is a challenge that has not been explored in this work. Although many of the language transfer patterns and characteristics that the string kernels approach is learning (e.g., morphological irregularities, lexical choice, among others) would also apply across genre, it is unclear whether such signals would be strong enough in shorter texts (like a single tweet). On the other hand, these models should be able to perform similarly well on other kinds of extended writing where there would be sufficient language transfer signals for the models to pick up on. These explorations will surely be addressed in future work, if such corpora become available.

6. Language Transfer Analysis

In order to better understand why these simple character p-gram models work so well, a focused manual analysis of the most discriminant features for each L1 is carried out next. The features that are about to be analyzed are learned from the TOEFL11 training and development sets, which contain 11,000 documents together. Following the analysis of Malmasi and Dras (2014c), character sequences that are overused or underused by a particular group of L1 speakers are considered to be discriminating. Although this analysis is carried out for English, a similar analysis for Norwegian and Arabic would of course be possible. This is out of scope of the current article, because none of the authors are experts in those languages.

As previously mentioned in Section 4, the features considered for the language transfer analysis were generated by the blended spectrum presence bits kernel of 5–9 p-grams, which was computed on the raw text documents. The dual weights α used in the analysis are learned by KRR using the one-versus-all scheme. According to the one-versus-all scheme, a binary classifier was built to discriminate each L1 from the others, resulting in a set of 11 dual weight vectors. Because the string kernels are based on the dual representation, the first step in order to determine which features are most discriminant, is to obtain the weights w corresponding to the primal representation. The high dimensionality of the explicit feature space can pose interesting challenges in terms of computational complexity and depending of the type of the kernel, elaborated and efficient solutions have been proposed in literature (Pighin and Moschitti 2009, 2010). Fortunately, in the case of the blended spectrum presence bits kernel of 5–9 p-grams, the dimensionality of the primal feature space (4,662,520) is manageable. This enables the direct reconstruction of the feature weights w from the dual weights α and the examples (represented in the primal space) using Equation (4). The primal weights of each binary classifier indicate both the overused and the underused patterns that help to discriminate a certain L1 language from the others. More precisely, the features that are associated with positive weights are those that are overused in the respective L1 language, whereas those that are associated with negative weights correspond to underused patterns. There are 4,662,520 features in total generated from the blended spectrum presence bits kernel, and each feature corresponds to a particular p-gram. It becomes clear that performing an analysis on this tremendous amount of features per native language is out of the question. Instead, the analysis presented in this work is focused only on the top (positive and negative) features that indicate which character sequences are most likely to help discriminate a particular L1 from the others. The

features that are not helpful for the classification task have weights close to zero and they are excluded from the analysis.

Table 12 lists interesting character sequences among the most discriminating overuse sequences for each L1 in the TOEFL11 corpus. Some interesting patterns can be observed here. First, although the corpus is not topic-biased, some of the most discriminant character sequences for many languages contain strings relating to country, language, or city names. This is evident from strings such as "german," "franc," "chin," "japan," and so on. Noticeably, for some L1s these strings do not appear in the top discriminant features. For example, Arabic and Spanish, which are spoken in many different countries, tend not to have country or city names as the top features. This may be because these writers talk about a wide range of different countries and cities and so the weights of any individual country feature might get demoted. Several teams in the 2013 NLI Shared Task also observed this result (Abu-Jbara et al. 2013; Gebre et al. 2013; Henderson et al. 2013; Malmasi, Wong, and Dras 2013; Mizumoto et al. 2013; Tsvetkov et al. 2013). Abu-Jbara et al. (2013) carried out a simple experiment where they only used features corresponding to country names and achieved an accuracy of 21.3% on the development set. Gebre et al. (2013) conducted a related experiment where they removed all country names from the set of features. They show that the accuracy of their system only drops 1.2 percentage points when these features are removed.

Second, there are some spelling errors particular to some sets of L1s that appear in the top features. For Spanish L1s, the spelling of "different" with only one "f" seems to be a common misspelling. Moreover, the missing space in "a lot" for Arabic L1s is quite discriminant, as is the misspelling of "their" as "thier." The misspelling of words such as "additionally" and "personally" that duplicates the "n" incorrectly seems to be quite a typical error for French native speakers. This is similar to some of the findings of Malmasi and Dras (2014c), who discover several misspellings as being typical for a particular group of L1 speakers. This is also a well-established phenomenon observed in work on second language acquisition and language transfer (Jarvis and Crossley 2012).

Other discriminant features indicating sentence-initial preferences can also be observed. For example, Chinese speakers are more likely to write a sentence that starts

Table 12
Examples of discriminant overused character sequences with their ranks (left) according to the KRR model based on blended spectrum presence bits kernel extracted from the TOEFL11 corpus (English L2).

German		French		Arabic		Hindi		Spanish		Chinese	
1	, that	1	indeed	1	alot	2	as compa	1	, is	2	t most
6	german	19	onnal	9	any	9	hence	2	difer	4	chin
11	. but	21	is to	13	them	16	then	13	, but	7	just
13	often	26	franc	16	thier	17	indi	15	, etc	8	still
207	special	28	to concl	19	his	21	towards	17	cesar	14	. take

Italian		Japanese		Korean		Telugu		Turkish	
1	ital	1	japan	1	korea	1	i concl	1	i agree.
3	o beca	15	. if	24	e that	6	days	11	turk
4	fact	19	i disa	27	. as	7	.the	21	. becau
9	, for	27	. the	30	soci	11	where as	32	s about
24	the life	38	. it	36	. also	13	e above	37	being

with "take" over all other L1s. Korean speakers tend to prefer starting a sentence with "as" or "also," and Japanese speakers more often start their sentences with "if." There is also discriminating use of adverbs. Similar to Malmasi and Dras (2014c), the overused patterns presented in Table 12 show that Hindi speakers tend to use "hence" more often than other speakers. It can also be observed that Chinese speakers tend to use "just" or "still" more often than other L1s, whereas German speakers are inclined to prefer the use of "often" or "special."

The discriminant features confirm the hypothesis that the model based on character *p*-grams is capturing information both about lexical items but also about morphology and grammatical preferences. For example, stems such as "german," which appear in context referring to Germany, Germans, German, Germanic are common in the list of discriminant features. On the other hand, a preference for a definite article before "life" can be noticed in Italian. Hints of grammatical usage tendencies can also be easily observed. For example, one of the most discriminant patterns for German speakers is that they include a comma before "that." This reflects the grammatical requirement of the equivalent "dass" clause in German. Hindi speakers most often use the phrase "as compared." French speakers like to use the phrase "To conclude" rather than "In conclusion" or something similar, and Turkish speakers prefer to have the complete sentence "I agree" without any complement clause.

In Table 13, some of the most negatively weighted character sequences for each language are listed. The assumption is that if a character sequence receives a high negative weight in the model, then it indicates that that character sequence is under-used by speakers of that language. Simply looking at the most negatively weighted features for a specific L1 only tells which character sequences are helpful in predicting that an essay does not belong to the respective L1. Many of the negatively weighted character sequences are common sequences shared by speakers of almost all languages, and they may even correspond to overused patterns in some other L1 language. Therefore, the analysis of underused patterns is restricted to the top 1,000 negatively weighted features, and for each L1, the character sequences that only appear in the top 1,000 for that one language are chosen. In this way, the character sequences that do not offer interesting insights are removed from the analysis.

Table 13
Examples of discriminant underused character sequences (ranks omitted for readability) according to the KRR model based on blended spectrum presence bits kernel extracted from the TOEFL11 corpus (English L2).

German	French	Arabic	Hindi	Spanish	Chinese
. for ex	cause	a lot	s will	after	reasons
true	. secon	.for	main	nd so	an a
s, and	, becaus	too	time.	especial	have to
keep	table	.they	deep	but now	if the
using	subjects	conclu	great	have dif	. even

Italian	Japanese	Korean	Telugu	Turkish
s. so	gree tha	in japan	. also	most ad
. and i	in my	talk	somet	thier
into	give	them.	maybe	. first,
. it is	eem much	full	how t	grow
. that	it can	both	focus	well

Some interesting patterns also emerge for the underused character sequences for each L1, including for example some pairs of character sequences. In Arabic the misspelling "alot" is a highly weighted feature, and correspondingly the correct spelling "a lot" receives a large negative weight in the model. The misspelling "thier" was overused by Arabic speakers, but underused by Turkish speakers. Also, the phrase "in japan" is most likely to be written by Japanese speakers, but noticeably very rarely by Korean speakers. There are also patterns of syntactic construction underuse, particularly in how different L1 speakers begin sentences. Germans tend to refrain from beginning a sentence with "for example," Turkish speakers seem to refrain from beginning sentences with "first," and Italian speakers refrain from starting sentences with "and I," "it is," or "that." The table also includes examples of disfavored lexical choices, including Chinese speakers underuse of "reasons," Telugu speaker underuse of "maybe," and Japanese speaker underuse of "give."

The features presented in Table 12 can be directly compared with the overuse category in Malmasi and Dras (2014c), where lexical features are listed for two languages, Arabic and Hindi. The second highest ranked feature for Arabic in Malmasi and Dras (2014c), "anderstand," roughly appears at rank 691 in the KRR and $\hat{k}^{0/1}_{5-9}$ model, and "mony" (rank 4 in their model) appears at rank 141. Interestingly, the models are finding similar features, however the ranking is different. Similarly, the features presented in Table 13 can be directly compared to the underuse category in Malmasi and Dras (2014c) (though it is unclear how they treated features with a high negative weight across several languages). The underuse of the determiner "an" in Chinese observed in Malmasi and Dras (2014a) can also be observed in Table 13.

7. Conclusions

A comprehensive overview and evaluation of a state-of-the-art approach to native language identification was presented in this article. The system works at the character level, making the approach completely language-independent, topic-independent, and linguistic-theory-neutral. The idea of combining the string kernels, either by kernel sum or by kernel alignment, proved to be extremely useful, producing the best NLI performance in all the experiments presented in this work. The best string kernels approach is 1.7 percentage points above the top scoring system of the 2013 NLI Shared Task. Furthermore, it has an impressive generalization capacity, achieving results that are 30 percentage points higher than the state-of-the-art method in the cross-corpus experiment. The best string kernels system also shows an improvement of 17.7 percentage points over the previous best accuracy on Arabic data (Malmasi and Dras 2014a), and an improvement of 14–16 percentage points over the results reported by Pepper (2012) on Norwegian data, proving that string kernels are indeed language independent.

Language transfer effects confirmed by analyzing the features selected by the string kernel classifier as being more discriminating were also discussed in this work. This analysis offered some clues as to why the string kernel approach is able to work so well and surpass state-of-the-art accuracy levels. In future work, string kernels can be used to identify the native language for different L2 languages, as long as public data sets become available. Furthermore, string kernels can be used as a tool to analyze language transfer effects on the respective L2 languages. There are, however, some limitations of this approach for the task of analyzing language transfer, since a limited number of local syntactic effects can ever be captured.

Acknowledgments

The authors thank the reviewers for their helpful comments in improving the quality of this work. The authors would also like to thank Shervin Malmasi and Mark Dras for evaluating their method on the ALC corpus using the leave one out procedure, which gave us the opportunity to adequately compare the results. The authors would further like to thank the researchers at the University of Bergen for their helpfulness in getting access to the Norwegian ASK corpus, and to Steve Pepper for his help in understanding the data. Moreover, the authors thank Keelan Evanini and Anastassia Loukina from ETS for all their helpful comments.

The work of Radu Tudor Ionescu was supported from the European Social Fund under grant POSDRU/159/1.5/ S/137750.

References

Abu-Jbara, Amjad, Rahul Jha, Eric Morley, and Dragomir Radev. 2013. Experimental results on the native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–88, Atlanta, GA.

Alfaifi, Abdullah, Eric Atwell, and Ibraheem Hedaya. 2014. Arabic Learner Corpus (ALC) v2: A New Written and Spoken Corpus of Arabic Learners. In *Proceedings of the Learner Corpus Studies in Asia and the World*, Kobe.

Blanchard, Daniel, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A Corpus of Non-Native English. Educational Testing Service Research Report No. RR–13–24. Princeton, NJ.

Bykh, Serhiy and Detmar Meurers. 2012. Native language identification using recurring *n*-grams—investigating abstraction and domain dependence. In *Proceedings of COLING*, pages 425–440, Mumbai.

Bykh, Serhiy and Detmar Meurers. 2014. Exploring syntactic features for native language identification: A variationist perspective on feature encoding and ensemble optimization. In *Proceedings of COLING*, pages 1962–1973, Dublin.

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

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

Dinu, Liviu P., Radu Tudor Ionescu, and Alexandru I. Tomescu. 2014. A rank-based sequence aligner with applications in phylogenetic analysis. *PLoS One*, 9(8):e104006, 08.

Dinu, Liviu P. and Florin Manea. 2006. An efficient approach for the rank aggregation problem. *Theoretical Computer Science*, 359(1–3):455–461.

Escalante, Hugo Jair, Thamar Solorio, and Manuel Montes-y-Gómez. 2011. Local histograms of character *n*-grams for authorship attribution. In *Proceedings of ACL: HLT*, 1:288–298, Portland, OR.

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

Gebre, Binyam Gebrekidan, Marcos Zampieri, Peter Wittenburg, and Tom Heskes. 2013. Improving native language identification with tf-idf weighting. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 216–223, Atlanta, GA.

Gonen, Mehmet and Ethem Alpaydin. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

Granger, Sylviane, 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.

Grozea, Cristian, 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, San Sebastian.

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

Haussler, David. 1999. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA.

Henderson, John, Guido Zarrella, Craig Pfeifer, and John D. Burger. 2013. Discriminating non-native English with 350 words. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 101–110, Atlanta, GA.

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

Ionescu, Radu Tudor. 2015. A fast algorithm for Local Rank Distance: Application to Arabic native language identification. In *Proceedings of ICONIP*, pages 390–400, Istanbul.

Ionescu, Radu Tudor, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language independent approach to native language identification. In *Proceedings of EMNLP*, pages 1363–1373, Doha.

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

Jarvis, Scott, Gabriela Castañeda Jiménez, and Rasmus Nielsen. 2004. Investigating L1 lexical transfer through learners' wordprints. *Second Language Research Forum (SLRF)*. State College, PA.

Jarvis, Scott and Scott Crossley, editors. 2012. *Approaching Language Transfer Through Text Classification: Explorations in the Detection-based Approach*, volume 64. Multilingual Matters Limited, Bristol, UK.

Kate, Rohit J. and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of ACL*, pages 913–920, Sydney.

Koppel, Moshe, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. In *Proceedings of ISI*, pages 209–217, Atlanta, GA.

Lodhi, Huma, 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.

Maji, Subhransu, Alexander C. Berg, and Jitendra Malik. 2008. Classification using intersection kernel support vector machines is efficient. In *Proceedings of CVPR*, pages 1–8, Anchorage, AK.

Malmasi, Shervin and Mark Dras. 2014a. Arabic native language identification. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 180–186, Doha.

Malmasi, Shervin and Mark Dras. 2014b. Chinese native language identification. In *Proceedings of EACL*, 2:95–99, Gothenburg.

Malmasi, Shervin and Mark Dras. 2014c. Language transfer hypotheses with linear SVM weights. In *Proceedings of EMNLP*, pages 1385–1390, Doha.

Malmasi, Shervin, Sze-Meng Jojo Wong, and Mark Dras. 2013. NLI shared task 2013: MQ submission. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–133, Atlanta, GA.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY.

Mizumoto, Tomoya, Yuta Hayashibe, Keisuke Sakaguchi, Mamoru Komachi, and Yuji Matsumoto. 2013. NAIst at the NLI 2013 shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 134–139, Atlanta, GA.

Pepper, Steve. 2012. Lexical transfer in Norwegian interlanguage: A detection-based approach. Master's thesis, University of Oslo, Olso.

Pighin, Daniele and Alessandro Moschitti. 2009. Reverse engineering of tree kernel feature spaces. In *Proceedings of EMNLP*, pages 111–120, Singapore.

Pighin, Daniele and Alessandro Moschitti. 2010. On reverse feature engineering of syntactic tree kernels. In *Proceedings of CoNLL*, pages 223–233, Uppsala.

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

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

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

Popescu, Marius and Radu Tudor Ionescu. 2013. The story of the characters, the DNA and the native language. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278, Atlanta, GA.

Rozovskaya, Alla and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*, pages 961–970, Cambridge, MA.

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

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

Swanson, Ben and Eugene Charniak. 2014. Data driven language transfer hypotheses. In *Proceedings of EACL*, pages 169–173, Gothenburg.

Tenfjord, K., P. Meurer, and K. Hofland. 2006. The ASK Corpus—A language learner corpus of Norwegian as a second language. In *Proceedings of LREC*, pages 1821–1824, Genoa.

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

Tetreault, Joel, 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*, pages 2585–2602, Mumbai.

Tomokiyo, Laura Mayfield and Rosie Jones. 2001. You're not from 'round here, are you? Naive Bayes detection of non-native utterances. In *Proceedings of NAACL*. Pittsburgh, PA.

Tsur, Oren and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 9–16, Prague.

Tsvetkov, Yulia, Naama Twitto, Nathan Schneider, Noam Ordan, Manaal Faruqui, Victor Chahuneau, Shuly Wintner, and Chris Dyer. 2013. Identifying the L1 of non-native writers: The CMU-Haifa system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 279–287, Atlanta, GA.

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