UnibucKernel: An Approach for Arabic Dialect Identification based on Multiple String Kernels

Radu Tudor Ionescu and **Marius Popescu**
University of Bucharest
Department of Computer Science
14 Academiei, Bucharest, Romania
raducu.ionescu@gmail.com
popescunmarius@gmail.com

Abstract

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. Unlike the common approach, we present a method that uses only character p-grams (also known as n-grams) as features for the Arabic Dialect Identification (ADI) Closed Shared Task of the DSL 2016 Challenge. The proposed approach combines several string kernels using multiple kernel learning. In the learning stage, we try both Kernel Discriminant Analysis (KDA) and Kernel Ridge Regression (KRR), and we choose KDA as it gives better results in a 10-fold cross-validation carried out on the training set. Our approach is shallow and simple, but the empirical results obtained in the ADI Shared Task prove that it achieves very good results. Indeed, we ranked on the second place with an accuracy of 50.91% and a weighted F_1 score of 51.31%. We also present improved results in this paper, which we obtained after the competition ended. Simply by adding more regularization into our model to make it more suitable for test data that comes from a different distribution than training data, we obtain an accuracy of 51.82% and a weighted F_1 score of 52.18%. Furthermore, the proposed approach has an important advantage in that it is language independent and linguistic theory neutral, as it does not require any NLP tools.

1 Introduction

It seems natural to use words as basic units in text categorization, authorship identification, plagiarism detection or similar text mining tasks. Perhaps surprisingly, recent results indicate that methods handling the text at the character level can also be very effective (Lodhi et al., 2002; Sanderson and Guenter, 2006; Kate and Mooney, 2006; Popescu and Dinu, 2007; Grozea et al., 2009; Popescu, 2011; Escalante et al., 2011; Popescu and Grozea, 2012; Ionescu et al., 2014; Ionescu et al., 2016). 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, we present a method based on character p-grams that we designed for the Arabic Dialect Identification (ADI) Shared Task of the DSL 2016 Challenge (Malmasi et al., 2016). In this task, the participants had to discriminate between Modern Standard Arabic (MSA) and 4 Arabic dialects, in a 5-way classification setting. A number of 18 teams have submitted their results on the final test set, and our team (UnibucKernel) ranked on the second place with an accuracy of 50.91% and a weighted F_1 score of 51.31%. Our best scoring system is based on combining three different string kernels via multiple kernel learning (MKL) (Gonen and Alpaydin, 2011). The first kernel that we considered is the p-grams presence bits kernel[1], which takes into account only the presence of p-grams instead of their frequency. The second kernel is the (histogram) intersection string kernel[2], which was first used in a text mining task by Ionescu et al. (2014), although it is much more popular in computer vision (Maji et al., 2008; Vedaldi and Zisserman, 2010). The third kernel is derrived from Local Rank Distance[3], a distance measure that

[1]We computed the p-grams presence bits kernel using the open source code provided at http://string-kernels.herokuapp.com

[2]We computed the intersection string kernel using the open source code provided at http://string-kernels.herokuapp.com

[3]We computed the Local Rank Distance using the open source code provided at http://lrd.herokuapp.com

Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects,
pages 135–144, Osaka, Japan, December 12 2016.

was first introduced in computational biology (Ionescu, 2013; Dinu et al., 2014), but it has also shown its application in NLP (Popescu and Ionescu, 2013; Ionescu, 2015). Although character p-grams have been employed for ADI in several works (Darwish et al., 2014; Zaidan and Callison-Burch, 2014; Malmasi et al., 2015), to the best of our knowledge, none of these string kernels have been previously used for ADI. Interestingly, these kernels have also been used for native language identification (Popescu and Ionescu, 2013; Ionescu et al., 2014; Ionescu et al., 2016), obtaining state-of-the-art performance for several languages, including Arabic.

Two kernel classifiers (Shawe-Taylor and Cristianini, 2004) were considered for the learning task, namely Kernel Ridge Regression (KRR) and Kernel Discriminant Analysis (KDA). The KDA classifier is sometimes able to improve accuracy by avoiding the masking problem (Hastie and Tibshirani, 2003). In a set of preliminary experiments performed on the training set, we found that KDA gives slightly better results than KRR. Hence, all our submissions are based on learning with KDA. Before submitting our results, we have also tuned our string kernels for the task. First of all, we tried out p-grams of various lengths, including blended variants of string kernels as well. The best accuracy was obtained with blended kernels of 3 to 6 p-grams. Second of all, we have evaluated the individual kernels and various MKL combinations. The empirical results indicate that combining kernels via MKL can help to improve the accuracy by nearly 1%. All these choices played a significant role in obtaining the second place in the final ranking of the ADI Shared Task. After the challenge, as we learned that the test set comes from a different source, we further improved our models just by adding more regularization. Interestingly, our approach treats the text documents simply as strings, since it does not involve any linguistic processing of the text, not even tokenization. Therefore, our method is language independent and linguistic theory neutral. Furthermore, the proposed approach is simple and effective, as it is just based on shallow features (character p-grams).

The paper is organized as follows. Work related to Arabic dialect identification and to methods based on string kernels is presented in Section 2. Section 3 presents the string kernels that we used in our approach. The learning methods used in the experiments are described in Section 4. Section 5 presents details about experiments, including parameter tuning, combining kernels and results of submitted systems. Finally, we draw our conclusion in Section 6.

2 Related Work

2.1 Arabic Dialect Identification

Arabic dialect identification is a relatively new NLP task with only a handful of works to address it (Biadsy et al., 2009; Zaidan and Callison-Burch, 2011; Elfardy and Diab, 2013; Darwish et al., 2014; Zaidan and Callison-Burch, 2014; Malmasi et al., 2015). Although it did not received too much attention, the task is very important for Arabic NLP tools, as most of these tools have only been design for Modern Standard Arabic. Biadsy et al. (2009) describe a phonotactic approach that automatically identifies the Arabic dialect of a speaker given a sample of speech. While Biadsy et al. (2009) focus on spoken Arabic dialect identification, others have tried to identify the Arabic dialect of given texts (Zaidan and Callison-Burch, 2011; Elfardy and Diab, 2013; Darwish et al., 2014; Malmasi et al., 2015). Zaidan and Callison-Burch (2011) introduce the Arabic Online Commentary (AOC) data set of 108K labeled sentences, 41% of them having dialectal content. They employ a language model for automatic dialect identification on their collected data. A supervised approach for sentence-level dialect identification between Egyptian and MSA is proposed by Elfardy and Diab (2013). Their system outperforms the approach presented by Zaidan and Callison-Burch (2011) on the same data set. Zaidan and Callison-Burch (2014) extend their previous work (Zaidan and Callison-Burch, 2011) and conduct several ADI experiments using word and character p-grams. Different from most of the previous work, Darwish et al. (2014) have found that word unigram models do not generalize well to unseen topics. They suggest that lexical, morphological and phonological features can capture more relevant information for discriminating dialects. As the AOC corpus is not controlled for topic bias, Malmasi et al. (2015) also state that the models trained on this corpus may not generalize to other data as they implicitly capture topical cues. They perform ADI experiments on the Multidialectal Parallel Corpus of Arabic (MPCA) (Bouamor et al., 2014) using various

word and character p-grams models in order to assess the influence of topic bias. Interestingly, Malmasi et al. (2015) find that character p-grams are "in most scenarios the best single feature for this task", even in a cross-corpus setting. Their findings are consistent with our results in the ADI Shared Task of the DSL 2016 Challenge (Malmasi et al., 2016), as we ranked on the second place using solely character p-grams. It is important to remark that the ADI Shared Task data set contains Automatic Speech Recognition (ASR) transcripts of Arabic speech collected from the Broadcast News domain (Ali et al., 2016). The fact that the data set may contain ASR errors (perhaps more in the dialectal speech segments) makes the ADI task much more difficult than in previous studies.

2.2 String Kernels

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 et al., 2009; Popescu, 2011; Escalante et al., 2011; Popescu and Grozea, 2012; Ionescu et al., 2014; Ionescu et al., 2016). 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. More recently, Ionescu et al. (2016) have used various blended string kernels to obtain state-of-the-art accuracy rates for native language identification.

3 Similarity Measures for Strings

3.1 String Kernels

The kernel function gives kernel methods the power to naturally handle input data that is not in the form of numerical vectors, 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). 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.

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

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

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

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

[4]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.

In computer vision, the (histogram) intersection kernel has successfully been used for object class recognition from images (Maji et al., 2008; Vedaldi and Zisserman, 2010). Ionescu et al. (2014) have used the intersection kernel as a kernel for strings. The intersection string kernel is defined as follows:

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

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

For the p-spectrum kernel, the frequency of a p-gram has a very significant contribution to the kernel, since it considers the product of such frequencies. On the other hand, the frequency of a p-gram is completely disregarded in the p-grams presence bits kernel. The intersection kernel lies somewhere in the middle between the p-grams presence bits kernel and p-spectrum kernel, in the sense that the frequency of a p-gram has a moderate contribution to the intersection kernel. In other words, the intersection kernel assigns a high score to a p-gram only if it has a high frequency in both strings, since it considers the minimum of the two frequencies. The p-spectrum kernel assigns a high score even when the p-gram has a high frequency in only one of the two strings. Thus, the intersection kernel captures something more about the correlation between the p-gram frequencies in the two strings. Based on these comments, we decided to use only the p-grams presence bits kernel and the intersection string kernel for ADI.

Data normalization helps to improve machine learning performance for various applications. Since the value range of raw data can have large variation, classifier objective functions will not work properly without normalization. After normalization, each feature has an approximately equal contribution to the similarity between two samples. To obtain a normalized kernel matrix of pairwise similarities between samples, 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-grams presence bits kernel and the intersection kernel are being used. Taking into account p-grams of different length and summing up the corresponding kernels, new kernels, termed *blended spectrum kernels*, can be obtained. We have used various blended spectrum kernels in the experiments in order to find the best combination.

3.2 Local Rank Distance

Local Rank Distance (Ionescu, 2013) is a recently introduced distance measure for strings that aims to provide a better similarity than rank distance (Dinu and Manea, 2006). Local Rank Distance (LRD) has already shown promising results in computational biology (Ionescu, 2013; Dinu et al., 2014) and native language identification (Popescu and Ionescu, 2013; Ionescu, 2015).

In order to describe LRD, we use the following notations. 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 $x[i]x[i+1] \cdots x[j-1]$.

Local Rank Distance 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 Local Rank Distance. Otherwise, the maximal offset m is added to the Local Rank Distance. An important remark

is that LRD does not impose any mathematically developed global constraints, such as matching the i-th occurrence of a p-gram in x with the i-th occurrence of that same p-gram in y. Instead, it is focused on the local phenomenon, and tries to pair equal p-grams at a minimum offset. To ensure that LRD is a (symmetric) distance function, the algorithm also has to sum up the offsets obtained from the above process by exchanging x and y. LRD is formally defined in (Ionescu, 2013; Dinu et al., 2014).

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 average length of the strings. We set $m = 500$ in our experiments, which is about twice the average length of the ASR transcripts provided in the training set. In the experiments, the efficient algorithm of Ionescu (2015) is used to compute LRD. However, LRD needs to be used as a kernel function. We use the RBF kernel (Shawe-Taylor and Cristianini, 2004) to transform LRD into a similarity measure:

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

where s and t are two strings and p is the p-grams length. The parameter σ is usually chosen so that values of $\hat{k}(s,t)$ are well scaled. In the above equation, Δ_{LRD} is already normalized to a value in the $[0,1]$ interval to ensure a fair comparison of strings of different length. Hence, we set $\sigma = 1$ in the experiments. The resulted similarity matrix is then squared in order to make sure it becomes a symmetric and positive definite kernel matrix.

4 Learning Methods

Kernel-based learning algorithms work by embedding the data into a Hilbert feature space, and searching for linear relations in that space. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. More precisely, a kernel matrix that contains the pairwise similarities between every pair of training samples is used in the learning stage to assign a vector of weights to the training samples. Let α denote this weight vector. In the test stage, the pairwise similarities between a test sample x and all the training samples are computed. Then, the following binary classification function assigns a positive or a negative label to the test sample:

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

where x is the test sample, n is the number of training samples, $X = \{x_1, x_2, ..., x_n\}$ is the set of training samples, k is a kernel function, and α_i is the weight assigned to the training sample x_i.

The advantage of using the dual representation induced by the kernel function becomes clear if the dimension of the feature space m is taken into consideration. Since string kernels are based on character p-grams, the feature space is indeed very high. For instance, using 5-grams based only on the 28 letters of the basic Arabic alphabet will result in a feature space of $28^5 = 17,210,368$ features. However, our best model is based on a feature space that includes 3-grams, 4-grams, 5-grams and 6-grams. 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 ADI experiments, we used the Kernel Ridge Regression (KRR) binary classifier. Kernel Ridge Regression selects the vector of weights that simultaneously has small empirical error and small norm in the Reproducing Kernel Hilbert Space generated by the kernel function. KRR is a binary classifier, but Arabic dialect 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:

	EGY	GLF	LAV	NOR	MSA
Train set	1578	1672	1758	1612	999
Test set	315	256	344	351	274

Table 1: The sample distribution per class for the ADI Shared Task training and test sets.

one-versus-all and one-versus-one. We considered the one-versus-all scheme for our Arabic dialect classification task. 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 ADI, the masking problem may appear, for instance, when an Arabic dialect A is somehow related to two other Arabic dialects 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. 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 dialects are related by any means, such as geographical position or quantitative linguistic analysis. More details about KRR and KDA are given in (Shawe-Taylor and Cristianini, 2004).

5 Experiments and Results

5.1 Data Set

The ADI Shared Task data set (Ali et al., 2016) contains Automatic Speech Recognition (ASR) transcripts of Arabic speech collected from the Broadcast News domain. The task is to discriminate between Modern Standard Arabic (MSA) and 4 Arabic dialects, namely Egyptian (EGY), Gulf (GLF), Levantine (LAV), and North-African or Maghrebi (NOR). Table 1 shows the sample distribution per class for the training and the test sets. As the samples are not evenly distributed, an accuracy of 22.79% can be obtained with a majority class baseline. Another important aspect is that the training and the test set are taken from different sources, and this could alter the performance of a classifier. However, we were unaware of this fact before the submission deadline.

5.2 Parameter Tuning and Implementation Choices

In our string kernels approach, ASR transcripts are treated as strings. Because the approach works at the character level, there is no need to split the texts into words, or to do any NLP-specific processing before computing the string kernels. The only editing done to the texts was the replacing of sequences of consecutive space characters (space, tab, and so on) with a single space character. This normalization was needed in order to prevent the artificial increase or decrease of the similarity between texts, as a result of different spacing.

In order to tune the parameters and to decide what kernel learning method works best, we fixed 10 folds in order to evaluate each option in a 10-fold cross-validation (CV) procedure on the training set. We first carried out a set of preliminary experiments to determine the range of p-grams that gives the most accurate results in the 10-fold CV procedure. We fixed the kernel method to KRR based on the Local Rank Distance kernel ($\hat{k}_p^{LRD}$) and we evaluated all the p-grams in the range 2-7. The results are illustrated in Figure 1. Interestingly, the best accuracy (64.97%) is obtained with 4-grams. Furthermore, experiments with different blended kernels were conducted to see whether combining p-grams of different lengths could improve the accuracy. More precisely, we evaluated combinations of p-grams in three ranges: 3-5, 4-6 and 3-6. In the end, the best accuracy (66.43%) was obtained when all the p-grams with the length in the range 3-6 were combined. Hence, we used blended kernels with p-grams in the range 3-6 in the subsequent experiments.

Further experiments were also performed to establish what type of kernel works better, namely the blended p-grams presence bits kernel ($\hat{k}_{3-6}^{0/1}$), the blended p-grams intersection kernel ($\hat{k}_{3-6}^{\cap}$), or the kernel based on LRD ($\hat{k}_{3-6}^{LRD}$). 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

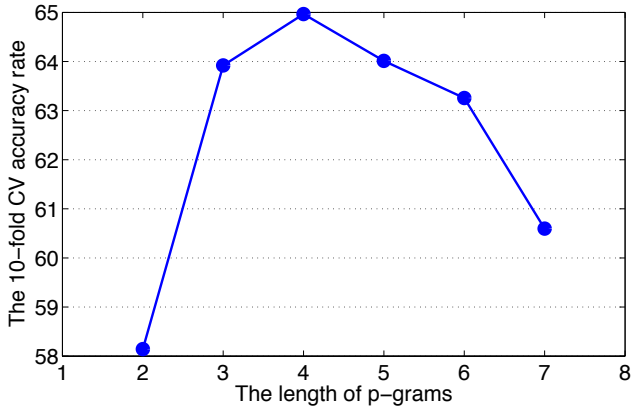

Figure 1: Accuracy rates of the KRR based on the LRD kernel with p-grams in the range 2-7. The results are obtained in a 10-fold cross-validation carried out on the training set.

Kernel	KRR	KDA
$\hat{k}^{0/1}_{3-6}$	65.89%	66.18%
$\hat{k}^{\cap}_{3-6}$	65.74%	66.28%
$\hat{k}^{LRD}_{3-6}$	66.43%	66.54%
$\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6}$	65.96%	66.42%
$\hat{k}^{0/1}_{3-6} + \hat{k}^{LRD}_{3-6}$	66.64%	67.17%
$\hat{k}^{\cap}_{3-6} + \hat{k}^{LRD}_{3-6}$	66.81%	67.12%
$\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6} + \hat{k}^{LRD}_{3-6}$	66.98%	**67.37%**

Table 2: Accuracy rates of different blended string kernels combined with either KRR or KDA. The results are obtained in a 10-fold cross-validation carried out on the training set. The best result is highlighted in bold.

Dialects	EGY	GLF	LAV	NOR	MSA
EGY	171	39	50	34	21
GLF	45	112	49	22	28
LAV	43	68	167	36	30
NOR	50	75	40	171	15
MSA	21	34	24	18	177

Table 3: Confusion matrix (on the test set) of KDA based on the sum of the blended p-grams presence bits kernel and the blended intersection kernel. The regularization parameter is set to 0.8, so the F_1 score of this model is 52.18%.

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. The kernels were evaluated alone and in various combinations, by employing either KRR or KDA for the learning task. All the results obtained in the 10-fold CV carried out on the training set are given in Table 2.

The empirical results presented in Table 2 reveal several interesting aspects about the proposed methods. Regarding the two kernel classifiers, it seems that KDA gives consistently better results, although the difference in terms of accuracy is almost always less than 0.5%. The individual kernels obtain fairly similar results. Perhaps surprisingly, the best individual kernel is the kernel based on Local Rank Distance with an accuracy of 66.43% when it is combined with KRR, and an accuracy of 66.54% when it is combined with KDA. Each and every kernel combination yields better results than each of its individual components alone. For both KRR and KDA, the best accuracy is actually obtained when all three kernels are combined together. Indeed, KRR reaches an accuracy of 66.98% when the blended p-grams presence bits kernel, the blended intersection kernel and the blended LRD kernel are summed up. With the same kernel combination, KDA yields an accuracy of 67.37%. As KDA gives consistently better results in the 10-fold CV procedure, we decided to submit three KDA models for the test set. The first submission (run1) is based on the LRD kernel, which seems to be the best one among the individual kernels, although previous works (Ionescu et al., 2014; Ionescu et al., 2016) indicate that the other two kernels obtain better results on native language identification. Influenced by these previous works, we also decided to give a fair chance to the blended p-grams presence bits kernel and the blended intersection kernel. Hence, the second submission (run2) is based on the sum between $\hat{k}^{0/1}_{3-6}$ and $\hat{k}^{\cap}_{3-6}$. Finally, our third submission (run3) is based on the sum of all three kernels, as this combination yields the best overall accuracy in the 10-fold CV procedure carried out on the training set.

Method	Reg.	Accuracy	$\mathbf{F_1}$ (macro)	$\mathbf{F_1}$ (weighted)	Submitted
KDA and $\hat{k}^{LRD}_{3-6}$	0.1	49.29%	49.43%	49.54%	Yes (run1)
KDA and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6}$	0.4	50.84%	51.09%	51.23%	Yes (run2)
KDA and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6} + \hat{k}^{LRD}_{3-6}$	0.2	50.91%	51.21%	51.31%	Yes (run3)
KDA and $\hat{k}^{LRD}_{3-6}$	0.2	49.35%	49.51%	49.59%	No
KDA and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6}$	0.8	51.82%	52.00%	52.18%	No
KDA and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6} + \hat{k}^{LRD}_{3-6}$	0.4	51.49%	51.52%	51.66%	No
KRR and $\hat{k}^{LRD}_{3-6}$	10^{-4}	50.19%	49.55%	49.72%	No
KRR and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6}$	10^{-4}	52.21%	51.73%	51.99%	No
KRR and $\hat{k}^{0/1}_{3-6} + \hat{k}^{\cap}_{3-6} + \hat{k}^{LRD}_{3-6}$	10^{-4}	51.88%	51.39%	51.56%	No

Table 4: Results for test set C (closed training) of various models based on string kernels. Some models that have not been submitted for the challenge are also included. For each model, the regularization parameter used to control the trade-off between overfitting and underfitting is reported as well.

5.3 Results and Discussion

Table 4 presents our results for the Arabic Dialect Identification Closed Shared Task (test set C) of the DSL 2016 Challenge, along with a few systems that were not submitted for the task. Among the three submissions, the best performance is obtained when all three kernels are combined and KDA is used for learning. The submitted systems were ranked by their weighted F_1 score, and among the 18 participants, our best model obtained the second place with a weighted F_1 score of 51.31%. Nevertheless, the winning solution is marginally better, with a difference of 0.0078% in terms of the weighted F_1 score.

A very important remark is that all our submitted systems obtain significantly lower results on the test set than in the 10-fold CV procedure carried out on the training set. This could be explained by the fact that the test set comes from a different distribution. As described by Ali et al. (2016), it actually seems that the training and the test sets come from different sources. In this context, regularization plays an important role, as it can be used to reduce the overfitting of the training data. A straightforward experiment, in which we simply double the regularization parameter of KDA, proves that all our submitted models yield better results when they are forced to fit less of the training data. Our best weighted F_1 score (52.18%) on the test set is obtained by the KDA based on the sum of the blended p-grams presence bits kernel and the blended intersection kernel. The confusion matrix of this model is given in Table 3. For this model, it takes about 12 minutes to compute the two kernels, train the KDA classifier and predict the labels on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM using a single Core.

Since the training and the test sets come from different distributions, the ADI task can also be regarded as a cross-corpus evaluation task. An interesting remark is that Ionescu et al. (2014) have used KRR and KDA in a cross-corpus setting for native language identification, and they have found that KRR is more robust in such a setting. Thus, we have also included results with KRR instead of KDA, while using the same kernels. The KRR based on the sum of the blended p-grams presence bits kernel and the blended intersection kernel is the best KRR model on the test set, with a weighted F_1 score of 51.99%.

6 Conclusion

We have presented a method based on character p-grams for the Arabic Dialect Identification (ADI) Shared Task of the DSL 2016 Challenge (Malmasi et al., 2016). Our team (UnibucKernel) ranked on the second place with a weighted F_1 score of 51.31%. As we learned that the training and the test sets come from different distributions (Ali et al., 2016), we were able to further improve our results after the challenge to a weighted F_1 score of 52.18%, which is better than the winning solution (51.32%). To improve the results even further, more advanced techniques suitable for the cross-corpus setting, such as semi-supervised or transfer learning, can be employed in future work.

Acknowledgments

The authors have equally contributed to this work. They thank the reviewers for helpful comments.

References

Ahmed Ali, Najim Dehak, Patrick Cardinal, Sameer Khurana, Sree Harsha Yella, James Glass, Peter Bell, and Steve Renals. 2016. Automatic dialect detection in arabic broadcast speech. *Proceedings of Interspeech*, pages 2934–2938.

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. 2009. Spoken Arabic Dialect Identification Using Phonotactic Modeling. *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 53–61.

Houda Bouamor, Nizar Habash, and Kemal Oflazer. 2014. A Multidialectal Parallel Corpus of Arabic. *Proceedings of LREC*, pages 1240–1245, may.

Kareem Darwish, Hassan Sajjad, and Hamdy Mubarak. 2014. Verifiably Effective Arabic Dialect Identification. *Proceedings of EMNLP*, pages 1465–1468.

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

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

Heba Elfardy and Mona T. Diab. 2013. Sentence Level Dialect Identification in Arabic. *Proceedings of ACL*, pages 456–461.

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

Mehmet Gonen and Ethem Alpaydin. 2011. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research*, 12:2211–2268, July.

Cristian Grozea, Christian Gehl, and Marius Popescu. 2009. ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. *Proceedings of 3rd PAN WORKSHOP*, page 10.

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

David Haussler. 1999. Convolution Kernels on Discrete Structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA.

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

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2016. String kernels for native language identification: Insights from behind the curtains. *Computational Linguistics*, 42(3):491–525.

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

Radu Tudor Ionescu. 2015. A Fast Algorithm for Local Rank Distance: Application to Arabic Native Language Identification. *Proceedings of ICONIP*, 9490:390–400.

Rohit J. Kate and Raymond J. Mooney. 2006. Using String-kernels for Learning Semantic Parsers. *Proceedings of ACL*, pages 913–920.

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

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

Shervin Malmasi, Eshrag Refaee, and Mark Dras. 2015. Arabic Dialect Identification using a Parallel Multidialectal Corpus. *Proceedings of PACLING*, pages 209–217, May.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and arabic dialect identification: A report on the third dsl shared task. *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (VarDial)*.

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

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

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

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

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

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

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

Omar F. Zaidan and Chris Callison-Burch. 2011. The Arabic Online Commentary Dataset: An Annotated Dataset of Informal Arabic with High Dialectal Content. *Proceedings of ACL: HLT*, 2:37–41.

Omar F. Zaidan and Chris Callison-Burch. 2014. Arabic dialect identification. *Computational Linguistics*, 40(1):171–202.