Kernel Methods and String Kernels for Authorship Analysis

Notebook for PAN at CLEF 2012

Marius Popescu[1] and Cristian Grozea[2]

[1] University of Bucharest, Romania
[2] Fraunhofer FOKUS, Berlin, Germany
popescunmarius@gmail.com cristian.grozea@brainsignals.de

Abstract This paper presents our approach to the PAN 2012 Traditional Authorship Attribution tasks and the Sexual Predator Identification task. We approached these tasks with machine learning methods that work at the character level. More precisely, we treated texts as just sequences of symbols (strings) and used string kernels in conjunction with different kernel-based learning methods: supervised and unsupervised. The results were extremely good, we ranked first in most problem and overall in the traditional authorship attribution task, according to the evaluation provided by the organizers.

1 Introduction

This paper presents our approach to the PAN 2012 Traditional Authorship Attribution tasks: closed-class / open-class attribution, authorship clustering / intrinsic plagiarism and Sexual Predator Identification task. We approached these tasks with machine learning methods that work at the character level. More precisely, we treated texts as just sequences of symbols (strings) and used string kernels in conjunction with different kernel-based learning methods: supervised and unsupervised.

Using words is natural in text analysis tasks like text categorization (by topic), authorship identification and plagiarism detection. Perhaps surprisingly, recent results have proved that methods handling the text at character level can also be very effective in text analysis tasks [7,12,10,4,9]. In [7] string kernels were used for document categorization with very good results. Trying to explain why treating documents as symbol sequences and using string kernels led to such good results the authors suppose that: "the [string] kernel is performing something similar to stemming, hence providing semantic links between words that the word kernel must view as distinct". String kernels were also successfully used in authorship identification [12,10]. A possible reason for the success of string kernels in authorship identification is given in [10]: "the similarity of two strings as it is measured by string kernels reflects the similarity of the two texts as it is given by the short words (2-5 characters) which usually are function words, but also takes into account other morphemes like suffixes ('ing' for example) which also can be good indicators of the author's style".

Even more interesting is the fact that two methods that obtained very good results for text categorization (by topic) [7] and authorship identification [10] are essentially

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

Using string kernels will make the corresponding learning method completely language independent because the texts will be treated as sequences of symbols (strings). Methods working at the word level or above very often restrict their feature space according to theoretical or empirical principles. For example, they select only features that reflect various types of spelling errors or only some type of words (function words) [1], etc.. These features prove to be very effective for specific tasks, but other, possibly good features, depending on the particular task, may exist. String kernels embed the texts in a very large feature space (all substrings of length k) and leave it to the learning algorithm (SVM or others) to select important features for the specific task, by highly weighting these features.

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

The rest of the paper is organized as follows: the next section presents the string kernels used, the following three sections: Authorship Attribution, Authorship Clustering and Sexual Predator Identification correspond to each major sub-task and describe for each of those the specific kernel method used, how it was applied and what were the results obtained. The paper ends with a conclusions section.

2 String Kernels

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

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

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

The kernel function offers to the kernel methods the power to naturally handle input data that are not in the form of numerical vectors, for example strings. The kernel function captures the intuitive notion of similarity between objects in a specific domain and can be any function defined on the respective domain that is symmetric and positive definite. For strings, many such kernel functions exist with various applications in computational biology and computational linguistics [13].

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:

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

where $\text{num}_v(s)$ is the number of occurrences of string v as a substring in s [3]. The feature map defined by this kernel associates to each string a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length p (p-grams).

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

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

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

Normalized versions of these kernels

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

ensure a fair comparison of strings of different lengths.

For the traditional authorship attribution tasks we have used a p-spectrum normalized kernel of length 5 ($\hat{k}_5$). We have chosen the length 5 because it was reported to work well in the case of document categorization [7] and authorship identification [10]. See also the comments from the introduction.

Because the p-spectrum kernel works at the character level, we didn't need to split the texts into words, or to do any NLP-specific preprocessing. The only editing done to the texts was the replacing of sequences of consecutive space characters (space, tab, new line, etc.) with a single space character. This normalization was needed in order to not artificially increase or decrease the similarity between texts as a result of different spacing. Also all uppercase letters were converted to the corresponding lowercase ones.

For sexual predator identification task the character p-grams presence bits kernel (normalized) of length 5 ($\hat{k}_5^{0/1}$) was used. The motivations for this choice are given in Section 5.

[3] Note that the notion of substring requires contiguity. See [13] for a discussion about the ambiguity between the terms "substring" and "subsequence" across different traditions: biology, computer science.

3 Authorship Attribution

3.1 Closed-Class Attribution

Closed-class attribution sub-task is a typical instance of a multiclass classification problem, the main particularity in this challenge being the small number of training examples available (2 per class).

In the area of kernel methods the dominant approach for classification is based on support vector machines (SVM). SVM was designed for binary classification problems. Nevertheless, SVM can be applied to multiclass classification problems by decomposing the multiclass problem into multiple binary classification problems using common decomposing schemes such as: one-versus-all and one-versus-one.

In our opinion these approaches to multiclass classification can turn the low number of training examples only into a deeper problem. For instance, applying the one-versus-one scheme in our case will generate binary classification problems with only 4 training examples. Over-fitting is then not only possible, but highly probable. For this reason we decided to use a kernel method that takes directly into account the multiclass nature of the problem.

We have chosen the kernel partial least squares (PLS) regression as the learning method [11,13]. PLS has received a great amount of attention in the field of chemometrics where the number of explanatory variables exceeds the number of observations, which is also exactly the situation in our case. PLS creates uncorrelated latent components (variables) which are linear combinations of the original predictor variables, this low-dimensional representation being then used in a linear regression. The construction of latent components depends on the predictor variables, but also depends on the response variable, the latent components being created by a process of maximizing the covariance among input and output variables. For details about the method see [11,13]. We have implemented the kernel PLS algorithm described in [13].

The method has one parameter which must be tuned, the number of latent components to be extracted (the number of iterations). The use of too small or too large dimensional models (number of components) can cause under-fitting respectively over-fitting. The number of components must be at most the rank of the training data matrix, so it must be smaller or equal to the number of training examples. Because the number of training examples was very small (2 examples per author) in all attribution problems we chose to use the maximum possible number of components, that is, in all classification problems we have set the number of components to be equal with the number of the training examples.

For closed-class attribution problems we have applied the PLS regression in the standard way (one-vs-all encoding): the output variable Y encodes the class of an example by a vector of -1 and +1, with +1 on the position corresponding to the example's class and for a new example the predicted label will be the position of the maximum of the predicted output $\hat{Y}$.

The decision of using kernel PLS regression for closed-class attribution problems was made after an empirical evaluation of the performance of the method and comparison of the results obtained by kernel PLS regression with that obtained by one-versus-all SVM and one-versus-one SVM.

Problem	PLS	SVM (ova)	SVM (ovo)	Best result in the competition
A	76.92%	**84.62%**	69.23%	**84.62%**
B	**53.85%**	38.46%	38.46%	**53.85%**
C	**100.00%**	88.89%	88.89%	**100.00%**
D	75.00%	50.00%	50.00%	**100.00%**
E	25.00%	25.00%	25.00%	**100.00%**
F	90.00%	90.00%	90.00%	**100.00%**
G	50.00%	50.00%	50.00%	**75.00%**
H	**100.00%**	33.33%	33.33%	**100.00%**
I	75.00%	50.00%	50.00%	**100.00%**
J	**100.00%**	50.00%	50.00%	**100.00%**
K	50.00%	50.00%	50.00%	**75.00%**
L	75.00%	75.00%	50.00%	**100.00%**
M	75.00%	75.00%	75.00%	**87.50%**
Overall	**72.75%**	58.48%	55.38%	70.61%

Table 1. The results obtained by kernel PLS regression, one-versus-all SVM, and one-versus-one SVM on the AAAC problems .

The very small number of training examples available for each competition problem made impossible a cross-validation evaluation and comparison. Fortunately, there is a data set very similar with the competition problems. This data set is "Ad-hoc Authorship Attribution Competition" (AAAC) data set [6]. AAAC data set consists of 13 problems that represents a wide variety of languages, lengths, and genres. Most of these problems are very similar with the competition problems, having many candidate authors and few examples per author for training.

We have used AAAC problems for an empirical evaluation of the performance of the kernel PLS regression, one-versus-all SVM and one-versus-one SVM. For all three methods we have used as kernel the p-spectrum normalized kernel of length 5 ($\hat{k}_5$). In the case of kernel PLS regression the number of latent components was set as described above to be equal with the number of training examples. The variant of SVM we have used was a hard margin SVM. Because problems have a small number of examples and a high dimensional feature space induced by the p-spectrum kernel, the data set is separable and the best working SVM is a hard margin SVM that finds the maximal margin hyperplane. Hard margin SVM has no parameter to be set.

The results obtained by kernel PLS regression, one-versus-all SVM, and one-versus-one SVM on the AAAC problems are given In Table 1. For reference, in the last column for each problem is listed the best result obtained for that problem by a system participating to the competition. Note that no single system obtained the best results for all problems. The best overall result obtained in the AAAC competition is that of the highest scoring participant.

These results motivated our choice of kernel PLS regression as our entry in the competition for closed-class attribution sub-task. In the competition, kernel PLS regression obtained the highest accuracy for all problems of the closed-class attribution sub-task. Table 2 presents the results obtained by our method for each problem in the closed-

Problem	PLS	SVM (ova)	SVM (ovo)
A	**100.00%**	**100.00%**	83.33%
C	**100.00%**	62.50%	50.00%
I	**92.86%**	78.57%	71.43%
Overall	**97.62%**	80.36%	68.25%

Table 2. The results obtained by kernel PLS regression, one-versus-all SVM and one-versus-one SVM for closed-class attribution sub-task problems

class attribution sub-task. For comparison in the table are also given the results that one-versus-all SVM and one-versus-one SVM would have obtained in the competition (the evaluation was made possible by the release of ground truth data). SVM with the encoding one-vs-all was better than SVM with one-vs-one, but PLS was better than both, as predicted.

3.2 Open-Class Attribution

For open-class attribution problems we need to decide when to predict a label and when not. What the kernel PLS regression returns is a vector $\hat{Y}$ of real values. In the closed-class case the predicted label was the position of the maximum of $\hat{Y}$. Starting with $\hat{Y}$, a standard way to decide when to predict is to establish a threshold and predict a label only when the corresponding maximum value of $\hat{Y}$ is above the threshold. This procedure can be problematic in our case because the very small number of training example imply a small number of latent components which can cause under-fitting phenomena. Indeed, there are cases when all values of $\hat{Y}$ are less than 0, but the maximum of $\hat{Y}$ still indicates correctly the label.

We have considered that what is important is the structure of $\hat{Y}$ not the actual values of $\hat{Y}$. If maximum of $\hat{Y}$ is far enough from the rest of the values of $\hat{Y}$ a prediction can be made, otherwise not. We have modeled "far enough" by the condition that the difference between the maximum of $\hat{Y}$ and the mean of the rest of the values of $\hat{Y}$ to be greater than a fixed threshold. To establish best value for this threshold we have computed the above statistic for all testing examples of the closed-class problems and have taken the value of the 20% quantile, 0.3333. Of course, it was possible to establish a separate threshold for each open-class problem based on the corresponding closed-class problem, but we preferred a unique threshold because in real life situations not every open-class task have a closed-class variant – so the approach we took is more realistic.

The results obtained with this procedure were good: 80% accuracy on the problem B, 76.47% accuracy on the problem D and 81.25% accuracy on the problem J.

These results suggest that the method deserves further investigation, especially of its connection with the outliers detection techniques.

4 Authorship Clustering

For this subtask we have chosen the spectral clustering [8,5] as the clustering method. Very briefly, the unnormalized spectral clustering algorithm works by first constructing a similarity graph from the data, computing the unnormalized Laplacian L of this graph and then finding the m eigenvectors $U_{n\times m}$ corresponding to the m smallest eigenvalues of L (ignoring the trivial constant eigenvector corresponding to the eigenvalue 0). Using a standard method like K-means, the rows of U are clustered yielding a clustering of the original observations. For more details and justification of the method the reader is referred to [8,5].

There are a number of details that one must take care of when applying spectral clustering in practice. One must choose how to compute the similarity between observations and how to transform these similarities into a similarity graph. As similarity between observations we have used the p-spectrum normalized kernel of length 5 ($\hat{k}_5$), the same as in the case of attribution. An important point in spectral clustering is to construct similarity graphs that reflect the local neighborhood relationships between observations. Starting from a similarity matrix, there are many ways to define a similarity graph that reflects local behavior. One of the most popular and the one that we have used is the mutual k-nearest-neighbor graph. In the case of the mutual k-nearest-neighbor graph, the parameter k, representing the number of nearest neighbors, must be set. We have built problems similar to E and F, and by examining the performance of spectral clustering on those with different values of k, we have eventually chosen $k = 12$.

We have focused our efforts to solving the problem in its initial formulation, as still given on the website of the competition:

> *in this problem you are given a text (which, for simplicity, is segmented into a sequence of "paragraphs") and are asked to cluster the paragraphs into exactly two clusters: one that includes paragraphs written by the "main" author of the text and another that includes all paragraphs written by anybody else.*

For each problem our method clusters the paragraphs into exactly two clusters, even though later when the test data were released some problems consisted of paragraphs written by more than two authors and the performance of a method on these problems is measured according to the respective number of clusters.

The method behaves very well on the problems having two clusters. The results obtained on these problems are given in Table 3.

Problem	**No. of paragraphs**	**Paragraphs correctly clustered**
Etest01	30	30 (100.00%)
Ftest01	20	20 (100.00%)
Ftest02	20	19 (95.00%)
Ftest03	20	16 (80.00%)
Ftest04	20	20 (100.00%)

Table 3. The results obtained by spectral clustering on the problems having two clusters

5 Sexual Predator Identification

5.1 Predators Identification

We have treated the sexual predator identification problem as a classification problem in which each chat participant contributes with precisely one example labeled as predator on not depending on how participant is considered. The contributions of each chat participant are joined into a single text, with line separators, all uppercase letters are converted to the corresponding lowercase ones, and consecutive white spaces are converted to a single space.

The resulted classification problem have been approached as in the previous cases with kernel based learning methods.

This time the kernel used was the character p-grams presence bits kernel (normalized) of length 5 ($\hat{k}_5^{0/1}$).

We have used 5-grams as an educated guess, as 5 is long enough to catch the ubiquitous " asl ", in general long enough to capture word stems in English, and short enough to warrant frequent-enough matches between related same-stem words. We have used this string kernel out of all possible alternatives again as an educated guess: it can be argued that identifying the predators based on the content of their chat logs is a semantic problem, therefore we have used the kernel that we know it works very well on a somewhat-related n-grams based semantic analysis problem, the network anomaly detection (often aimed at network intrusion detection). There the "signatures" are exposing the presence of the network attacks, here we hoped similarly something from what the predators write will act as a signature of their exploiting attempt. Using presence bits brings resilience to attempts of hiding the important part, the signature, by adding large quantities of normal content. Another shared feature of the predators identification problem and of network intrusion detection is the severe class unbalance (here only 142 out of the 100,000 chatters were predators).

Choosing the kernel methods and evaluating them was a difficult task taking into account the dimensions of the problem.

The resources we had at our disposal were: a 4 GB RAM dual-core machine and a 6 GB 4-core machine

Note 1. While we thank here without naming them to the ones who promised us help in getting access to better computational resources, those promises failed to ever materialize – in one case, and materialized much too late – in the other case. Therefore the major technical difficulty we had was processing 80 GB kernel matrices in 4 to 6 GB RAM computers. The solution we have found for this is blocking, using matrix chunks, blocks of 5,000x5,000 elements written as independent units on disk. While this works, it works as expected very slowly, therefore we were severely limited in the set of machine learning algorithms that we could attempt.

We have computed the nearest 10 labeled neighbors for each element, by computing the top 10 labeled neighbors in each row of 5000x5000-sized blocks, then doing a post-processing sort of the top 10 in each block. This procedure is guaranteed to produce at least 10 correct nearest neighbors. 10 was chosen as a compromise between our need to analyze the data and our limited resources.

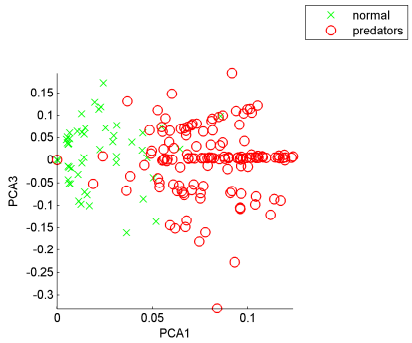

Figure 1. Classes distribution in the chosen representation; the lower-dimensional plot is obtained by retaining the three most important PCA projections

.

We have experimented with two methods: one based on k-Nearest Neighbors and the other based on random forests.

Method 1: k-NN (k-Nearest Neighbors) Simple statistics indicate that k-NN [3] on above representation has good chances to produce results for this problem; on the training dataset, after excluding the first neighbor (which is most of the time itself), in average 7.4859 of the first 9 neighbors of the predators are predators, whereas the mean of this statistic over the entire population is just 0.0479. We have used a variation of k-NN where we require that the first k neighbors of one sample being labeled as predators, in order to label the yet unlabeled sample as predator too. For choosing the best k we have tuned on the training data, and for k=4 we have achieved the best F-measure, 0.64 (precision=0.64, recall=0.64).

Method 2: Random forests After exploratory data analysis, consisting in examining plots of the data after various processing, we have used for our second (and better) submission random forests [2] based on a slightly different data representation: the similarity with the predators within the first 8 neighbors – from the first 8 rows of the kernel matrix; the similarity values for the non-predator neighbors are set to 0. Please note that the previous representation can be computed from this one (by ignoring the precise similarity values and counting only the non-zero values) which contains therefore more information.

The plot of the data in Figure 1 shows that this representation not only separates fairly nicely the predators from the normal population, but suggests also the existence of a structure / clusters in the set of predators.

Random forests method provides on the training dataset a different balance of about the same recall (0.67) and higher precision (0.77), for seemingly even higher F-measure

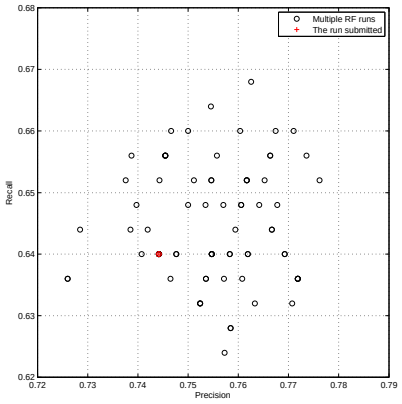

Figure 2. Stability of the accuracy of the RF solution measured on the test data

.

(0.72). The results on the test dataset were close to that: recall=0.64, precision=0.74, F-measure=0.69. As a note concerning the use of random forest on this classification, we look here at the variability of the prediction accuracy due to the implied in the building of the random forests model. As shown in Figure 2 the performance to be expected on the test set is fairly stable, despite the randomness implied in the random forests training.

To convince the reader that the n-grams binary embedding (with bits of presence) leads to a better kernel for this problem than the spectrum kernel, for n-grams of the same length (5), we have recomputed after the competition the kernel matrix, using the normalized spectrum kernel (which, we remind, uses the frequency of the n-grams in embedding, instead of their mere presence). The 5-fold cross-validation results for the same 100 trees random-forest model were as we had expected, considerably worse: recall=0.49, precision=0.76, and thus F-measure=0.60.

Interestingly, when raising the number of neighbors considered from 8 (used in our solution because of the very limited computing resources available to us at the time of this competition) to 16, there is a slight increase in the performance: recall=0.52, precision=0.79, F-measure=0.63.

One could conclude here that indeed the binary kernel is simply better than the spectrum kernel for this type of problems. Still, there is something interesting about the recall achievable with k-NN when using the spectrum kernel. As shown in figure 3, when deciding only on the first neighbor (1-NN), the recall is close to 0.9

Leveraging the capability of random forests to employ just the information contained in the useful inputs and to a large extent to ignore the useless / noise-containing ones, we have tried also to combine the two kernels instead of deciding which one is the best and selecting one. And indeed, by using as inputs to the machine learning problem the similarities to the predators in the first 8 neighbors for each of the two kernels, the 5-fold cross-validation performance increased sensibly over our best so far, show-

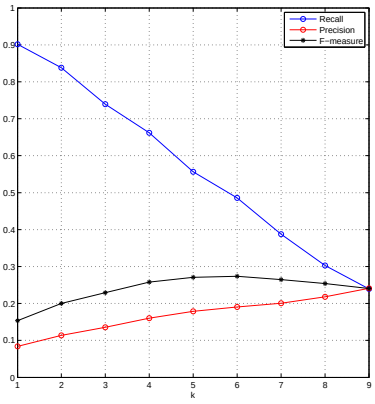

Figure 3. Balancing precision and recall with k-NN when using the spectrum kernel

.

ing once again the benefits of multiple kernel learning: recall=0.75, precision=0.83, F-measure=0.79.

5.2 Predators Lines Identification

There was also a second sub-task, where the very lines that prove the sexual predator's quality were to be found. As this was compulsory, we had to send something. A cursory examination of some of the training examples has shown us that there was no word wasted in their attempt to get alone with the future victim as soon as possible. Even the apparently innocent parts of the chat served a purpose. Therefore, we chose to simply mark as suspect *all* chat lines of the users we have identified (correctly or wrongly) as predators. Surprisingly this brought us a very good position (2^{nd}!) with very high recall and poor accuracy, position that eventually turned into 1^{st} after more after-the-fact tinkering with the scoring functions by the organizers (see the authorship clustering issue as well).

6 Conclusion

We have presented in this article our approaches to a rather diverse set of problems: the style-oriented closed-class and open-class attribution, authorship clustering / intrinsic plagiarism and the semantically-oriented sexual predators identification. As a result, we have used a variety of machine learning methods: kernel partial least squares (PLS), spectral clustering and random forests. What is shared in all these is our language-independent NLP-techniques light approach of building string kernels based on character-level n-grams (either spectrum or its binary embedding correspondent, where the frequencies are replaced by presence bits), followed by the application of state of the art

machine learning techniques. It is interesting to note that we have chosen the ML techniques not as much through extensive evaluation on the few data available for training as by their intrinsic qualities in what concerns the probability of over-fitting and the known success in parallel fields (such as chemometrics or computer networks security). The results were extremely good, we ranked first in most problems and overall in the traditional authorship attribution task, according to the evaluation provided by the organizers.

References

1. Argamon, S., Juola, P.: Overview of the international authorship identification competition at pan-2011. In: Petras, V., Forner, P., Clough, P.D. (eds.) CLEF (Notebook Papers/Labs/Workshop) (2011)
2. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
3. Cover, T., Hart, P.: Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13(1), 21–27 (1967)
4. Grozea, C., Gehl, C., Popescu, M.: ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In: 3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE. p. 10 (2009)
5. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. Springer, 2 edn. (2009)
6. Juola, P.: Authorship attribution. Foundations and Trends in Information Retrieval 1(3), 233–334 (2006)
7. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.J.C.H.: Text classification using string kernels. Journal of Machine Learning Research 2, 419–444 (2002)
8. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
9. Popescu, M.: Studying translationese at the character level. In: Proceedings of the International Conference Recent Advances in Natural Language Processing 2011. pp. 634–639. RANLP 2011 Organising Committee, Hissar, Bulgaria (September 2011)
10. Popescu, M., Dinu, L.P.: Kernel methods and string kernels for authorship identification: The federalist papers case. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-07). Borovets, Bulgaria (September 2007)
11. Rosipal, R., Trejo, L.J.: Kernel partial least squares regression in reproducing kernel hilbert space. Journal of Machine Learning Research 2, 97–123 (2001)
12. Sanderson, C., Guenter, S.: Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. pp. 482–491. Association for Computational Linguistics, Sydney, Australia (July 2006)
13. Taylor, J.S., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)