Author Verification: Exploring a Large set of Parameters using a Genetic Algorithm - Notebook for PAN at CLEF 2014

Erwan Moreau, Arun Jayapal, Carl Vogel

Author Verification: Exploring a Large set of Parameters using a Genetic Algorithm

Notebook for PAN at CLEF 2014

Erwan Moreau[1], Arun Jayapal[1], and Carl Vogel[2]

[1] CNGL and Computational Linguistics Group
moreaue@cs.tcd.ie, jayapala@cs.tcd.ie
[2] Computational Linguistics Group
vogel@cs.tcd.ie
Centre for Computing and Language Studies
School of Computer Science and Statistics
Trinity College Dublin
Dublin 2, Ireland

Abstract In this paper we present the system we submitted to the PAN'14 competition for the author verification task. We consider the task as a supervised classification problem, where each case in a dataset is an instance. Our system works by applying the same combination of parameters to every case in a dataset. Thus, the training stage consists in finding an optimal combination of parameters which maximizes the performance on the training data using cross-validation. This is achieved using a simple genetic algorithm, since the space of all possible combinations is impractical.

1 Introduction

In this author verification task, a training set containing 6 datasets was provided; each dataset consists of a set of problems (between 96 and 200) which belong to the same language and genre; each problem consists of a small set (between 1 and 5) of "known" documents written by a single person and a "questioned" document: the task is to determine whether the questioned document was written by the same person. More precisely, the system must provide its prediction as a value in the interval $[0, 1]$, which represents the probability that the answer is positive (same author). The intended interpretation is for 0 to mean "different author" with maximum certainty, and for 1 to mean "same author" with maximum certainty, and any intermediate value describes the likeliness of a positive answer, and with 0.5 equivalent to the system saying "I don't know". The predictions are evaluated using the product of the area under the ROC curve (AUC) and the modified accuracy measure c@1 [6], which treats 0.5 answers as a particular case.

We consider the task as a supervised learning problem, where, for each dataset, the goal is to find a function which, when applied to a set of unseen problems in this dataset, maximizes the performance (product of AUC and C@1). This function must be generic enough to capture the stylistic characteristics of every author. It is meant to represent *how* to capture any author's style within a particular dataset, that is, in the

context of a particular language and genre. For example, the type of observations (e.g., word bigrams) to take into account depends on the language, but whether a particular observation (e.g., the bigram "it is") is relevant or not is specific to a given author.

We define the space of all possible functions in the following way: each function is defined by a set of parameters, each parameter being assigned a particular value among a predefined set of possible values. The process of selecting features from the texts, combining them in any predefined way, and learning how to interpret the differences between the known documents and the questioned document is entirely driven by the values taken by these parameters. For example, a parameter indicates which distance metric should be used to compare the unknown document to the set of known documents. We call a particular combination of parameters a *configuration*. We define the two following strategies which share only a subset of common parameters:

- The *fine-grained strategy*, described in §3, in which there are many possible parameters, is intended to try as many configurations (or functions) as possible, in order to maximize the performance.
- The *robust strategy*, described in §4, is a more simple method which uses only a small subset of parameters. It is intended to be safer (in particular less prone to overfitting), but probably not to perform as well as the fined-grained strategy.

For the fine-grained strategy, the space of all possible configurations is too big to be explored exhaustively. This is why we implement a simple genetic algorithm, which is supposed to converge to a (possibly local) optimal configuration. This algorithm is described in §3.4.

2 General architecture

For every problem to solve, we extract *observations*[3] from the set of known texts, and try to measure the four following abstract characteristics:

- their *consistency*, i.e. how consistently these observations apply among the known documents written by the author;
- their *divergence* (from a reference corpus), i.e. how much the frequency of these observations differs from the reference corpus (see §3.2);
- the *confidence* of the system in the reliability of these observations;
- the *distance* between the known documents and the questioned document with respect to these observations.

We consider multiple ways to compute and use these four characteristic values. In particular, the configuration file defines:

- the types of observations to take into account;
- the method to compute every characteristic at the observation level;
- the method to extract the most relevant subset of observations;

[3] We use the term "observation" here to avoid any confusion with the features used in the machine learning stage.

– the method to obtain a global value for each of the four characteristics;
– which subset of these values will be used as features in the machine learning stage.

The final step consists in training (or applying) a ML model based on these features. There can actually be two models: the first and most important one predicts the scores for each case in the dataset; the second optional one is meant to detect the ambiguous cases, so that they can be assigned 0.5 instead of their predicted score, in order to maximize the c@1 score.

In the *robust strategy* the parameters are restricted to a small set of possible configurations, whereas with the *fine-grained strategy*, on the contrary, we explore a vast space of parameters (about 10^{19} possible combinations in the predefined space that we use). This is why the learning stage for the latter consists in learning an optimal configuration using a genetic algorithm.

3 The fine-grained strategy

3.1 Observations and frequency statistics

We consider a large set of observations types, among which the configuration can use any subset. These are typically various kinds of n-grams, but not only:

– words (actually tokens) unigrams to trigrams;
– Characters trigrams to pentagrams.
– Part-Of-Speech (POS) tags unigrams to tetragrams;
– Combinations of POS tags and tokens, including skip-grams, e.g.: "`<POS tag> <token> <POS tag>`" or "`<token> _ <POS tag>`";
– "stop words" n-grams, i.e. tokens n-grams considering only a predefined list of the the most frequent tokens in the language,[4] from trigrams to pentagrams;
– Token length classes, where the tokens are classified depending on their length into 6 categories: lower than 2, 3 to 4, 5 to 6, 6 to 8, 8 to 10, more than 10.
– Token-Type Ratio (number of distinct tokens divided by total number of words).

The POS tags are computed using TreeTagger[5] [8]. The lists of most frequent words in the language are computed from the complete set of documents in the training data: we consider the 200 most frequent words, except for Dutch (100 most frequent words).

A few thresholds are applied when extracting these observations, in order to remove some noise in the data and/or improve efficiency:

– Minimum absolute frequency in a document (possible values: 2, 3, 5);
– Minimum proportion of documents among the reference corpus which contain the observation (possible values: 10%, 25%, 50%);
– Minmimum proportion of known documents containing the observation (only for known documents) (possible values: 30%, 51%);

[4] Other tokens are replaced with a special symbol, e.g. "`the _ _ is _`".
[5] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger. POS tags are not used for the Greek dataset.

The frequency of all the observations which fulfill these conditions relativized to the total possible number of such observations lpq is stored for very observation type specified in the configuration. For every observation, various statistics are computed based on the set of frequencies extracted from the known documents: mean, standard deviation, median, etc. Practically, in the training stage, the observations and the statistics are computed only once and then stored, so that the data can be used as many times as necessary with different combinations of parameters.

3.2 Features

The consistency, divergence, confidence and distance values are based on the frequency statistics extracted during the first step. At first they are computed for every distinct observation. Then they can be "synthetized" in different ways according to the configuration; the final features can be either specific to each observation type or global.

Consistency The consistency value is meant to represent how constant the use of a particular observation is, so that it can be assessed whether the observation is used in a similar way in the unknown document. For example, the standard deviation of the (relative) frequency of the observation among the known documents is a valid indicator (the lower it is, the higher the consistency is). Other statistics are available, e.g. range between minimum and maximum, ratio between first and third quartile, etc. However these statistics are more reliable with a high number of known documents, and require at the very least two distinct documents.

The goal of the consistency measure is to distinguish as far as possible between the observations which are specific to the author and those which are only specific to the document (for example the subject of an essay). This is why the more known documents there are the most accurate the consistency is. Consequently, cases which contain only one known document are irrelevant for consistency.[6]

Divergence The divergence measure is meant to represent to what extent a particular observation is specific to an author. This value is calculated against a reference corpus, which should ideally be an independent set of documents in the same language and genre as the dataset. But since we do not have access to such a corpus for every dataset, we simply consider the whole set of documents (known and unknown) in the training set as the reference corpus. Because it is meant only to measure divergence, the only important assumption that we make is that it contains documents written by a sufficient number of different authors, and that it is not massively imbalanced (for instance if most of the documents were by the same author).[7]

[6] It is possible then to use different parts of the document, but this is not reliable in general since the distinction between document specific and author specific observations cannot be made.

[7] Although it is quite unlikely given the size of the training sets, we do not have any guarantee that the second condition is satisfied in all the datasets provided. Assuming these conditions are fulfilled, the fact that the reference corpus contains documents by the author of the problem studied is not an issue, because it is frequent that a particular stylistic feature can be observed

The system can use different methods to measure the divergence of an observation: several simple statistics like the absolute difference between the frequency means (or medians), but also more complex measures which try to estimate the difference between the two distribution (known documents and documents in the reference corpus). Some of these measures assume a normal distribution of the observation frequency accross the documents:[8] we use several measures based on the Bhattacharrya distance [1]. These measures are also more reliable when the number of distinct documents is high.

Confidence The confidence measure is intended to find the most discriminative observations, based on their consistency and divergence values. Thus, it simply combines these two values in order to rank the observations by their discriminative power for the given author: for instance, an observation which is very consistent but not distinctive (or the opposite) might be less interesting than another one which is less consistent but has a better divergence value.

We consider various ways to combine the two numerical values: product, mean, geometric mean, weighted product, etc. There is also an option to ignore the consistency score (i.e. use the divergence as confidence), and another one for ranking the two values, so that the rank is used instead of the actual score.

Distance Finally the distance measure is meant to capture how different the unknown document is from the set of known documents. The distance value is usually not meaningful at the level of the observation, but becomes meaningful only once computed on a selected set of observations.

Various standard similarity or distance measures can be used, like the cosine or Jaccard similarity, but also some more specific measures, like the probability that the observed frequency in the unknown document belongs to the distribution observed in the known documents (assuming this is a normal distribution). The distance can be weighted in different ways with a coefficient based on the confidence score.

3.3 Scoring stage

The configuration defines what kind and how many features will be used, as well as how to obtain them. Multiple possibilities have been implemented, including:

- There can be a set of features for each observations type, or all observations types can be combined in a generic set of features;
- The maximum number of observations to consider for the distance feature(s);
- Whether consistency, divergence and confidence scores should be included in the features.

with several authors: the relative ordering of the observations according to their specificity to an author should not be impacted.

[8] We had observed in [5] that this assumption holds in most cases for frequent n-grams.

A regression model is trained/applied to the features which have been computed for all the input cases (instances for the model).[9] We use the Weka [3] (version 3.6.10) implementation of SVM regression [4] (with polynomial or RBF kernel), and decision trees regression [7], with variants depending on their parameters.

Optionally, a second model can be generated/applied in order to evaluate the confidence in each answer, and possibly replace it with 0.5 (unanswered case). This classification "confidence model" can use any of the available features, as well as the score computed with the first regression model.[10]

3.4 Genetic learning

The complete "author verification model" which is returned at the end of the training stage consists of the scoring model (that is, the regression model and optionally the confidence model), but also the configuration which was found to be optimal on the training set by cross-validation. This is achieved using the generic genetic algorithm described next.

The individuals in a population are the configurations, in which every parameter is assigned a particular value among a predefined set (the "genotype"). Starting from a random population, the algorithm iterates through each generation by selecting a proportion of the population to "breed" the next generation.

The method for making a configuration which performs better more likely to get selected is as follows: all the configurations in the population are ranked from 1 to N by their performance in ascending order. The probability of an individual being selected is defined as $r/N \times p \times 2$, where r is its rank and p is the proportion to retain as breeders. For example, if the population $N = 200$ and $p = 10\%$ (that is, 20 breeders are selected at each stage), the probability for the best performing configuration (with relative rank $N/N = 1$) to be selected is $1 \times 10\% \times 2 = 20\%$, and the probability for the worst performing configuration (with relative rank $1/200 = 0.05$) is 0.01. Since the average relative rank r/N is 0.5 by definition, the method selects, on average, $N \times 0.5 \times p \times 2 = N \times p$ breeders, as expected (consequently p must not be higher than 0.5).

Every new individual is generated based on two "parents" picked randomly among the breeders; every of its parameter is defined as one of its parents value (each having a 0.5 probability to be picked), but can be "mutated" with a (small) predefined probability. We also use two variants: one consists in reusing a few of the previous best individuals in each new generation (elitism), and the other in including a small proportion of totally random individuals.

4 The robust strategy

In the robust strategy, consistency and divergence features were used to verify whether the document X has been authored by the author of the given documents $Y = \{y_1, y_2, ...y_n\}$,

[9] When training the model, the Y/N answers are converted to 1/0, so that the predictions of the system are values in $[0, 1]$, which is the expected output format.

[10] In the learning stage, the model which was trained is applied to the instances. Depending on the configuration, the instances can be split up so that the second model is based on unseen instances (but then less instances are used to train each model, of course).

but in a slightly different way as above: the consistency defines how well the words or n-grams or character-grams were used consistently used across all the documents Y and X. Whereas, divergence defines how well document X is distinct from documents Y and viceversa. The intuition behind using this feature is that these features could provide an insight into how the document X and documents Y co-vary linguistically.

Divergence Motivated from the Jaccard similarity, we use a slight variant to compute the divergence of documents Y to document X (J_1) and of document X to documents Y (J_2):

$$J_1 = \frac{(p+q)}{(p+q+r)} \qquad J_2 = \frac{(p+r)}{(p+q+r)} \tag{1}$$

where p is the number of words found in both X and Y documents, q is the number of words found in Y but not in X and r is the number of words found in X but not in Y documents. J_1 will provide a measure on how distinct Y is from X, whereas J_2 will provide a measure on how distinct X is from Y.

The above provided are the document level metrics, which are used to compute the word-level divergence for X and Y. One assumption considered here for word-level metric; to compute divergence of word x_i in X to Y, when the word w_i is identified in Y, we assign a boolean value 0 assuming no divergence and when w_i is not identified in Y, we assign 1 to a temporary variable F assuming complete divergence of word w_i. With, F, J_1, J_2 and relative frequency values (rf_i^1 and rf_i^2) for each word, we compute the divergence for words in X to Y (d_{i,J_1}) and Y to X (d_{i,J_2}) as:

$$d_{i,J_1} = F * J_1 * rf_i^1 \qquad d_{i,J_2} = F * J_2 * rf_i^2 \tag{2}$$

Consistency Consistency is defined as the difference between the relative frequencies:

$$c_{i,J_1} = rf_i^1 - rf_i^2 \qquad c_{i,J_2} = rf_i^2 - rf_i^1 \tag{3}$$

These measures are based only on the characters tetragrams' frequencies (the other observations types are not taken into account). In order to train or apply the model, the scoring stage defined in the fine-grained strategy is used.

5 Observations and results

5.1 Genetic learning process

Since the system was being implemented specifically for the task and we had to deal with the time constraints for the competition, the process of tuning the genetic learning parameters was not carried out in optimal conditions. In particular, we could not afford to run many different cases, especially cases which require a long time; moreover, bugs were fixed and features were added along the process. This is why we are not able to provide here a very detailed analysis of the impact of the parameters on the evolution of the performance. Yet we did several preliminary tests in order to determine a couple

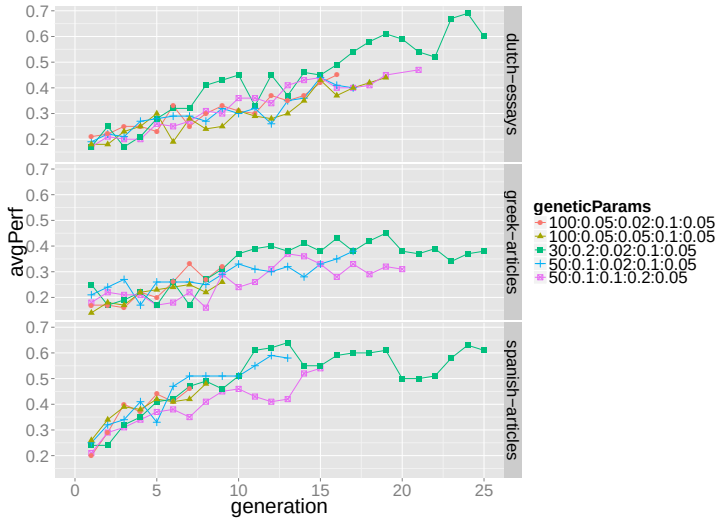

Figure 1. Average performance for various combinations of genetic parameters (preliminary test stage). *GeneticParams* contains the following parameters: population, breeders proportion; mutation probability; elitism proportion; random proportion. Example: on the longest curve (in green with square symbols), it can be observed that the average performance at the 20th generation was close to 0.6.[12]

of optimal combinations of genetic learning parameters. Figure 1 shows the results of one of these.

In general, the system was able to converge relatively quickly (a few tens of generations at most) to a high level of performance with most tested combinations of parameters. In particular, the size of the population did not have a major impact on the convergence (even though a larger population makes the evolution smoother). This is why we opted for using a small population size, in order to minimize the time required for the system to find optimal configurations. Figure 2 shows how the two selected sets of parameters performed during the main learning stage.[13] In total, between 13,400 and 26,700 configurations (in a space of 10^{19} possibilities) by dataset were evaluated in

[12] Moreover, the legend shows that this value is the average over a population of 30 configurations obtained after 19 iterations, where, at each stage: 20% of the previous configurations are selected as breeders (i.e., 6 configurations); the probability of a mutation (of an individual parameter) is 0.02; 10% (i.e., 3 configurations) of the new generation is made of the 10% best previous configurations ("cloned" directly without any alteration); 5% (i.e., 1 or 2 configurations) are totally random configurations.

[13] A recent server (24 Intel Xeon 3GHz cores) was used for the computation, but with only one core for each pair dataset/genetic configuration. It is difficult to evaluate exactly the total time spent due to various technical interruptions. Computing a single generation took in average between 20 and 48 minutes (depending mostly on the size of the dataset) for the "fast" configuration (30 individuals) and between 50 and 117 minutes for the "slow" configuration (75 individuals).

the genetic learning process. Each configuration was evaluated using only 3 fold cross-validation during the main genetic process; after this stage, a subset of the best configurations found was evaluated again using 10 fold and then 20 fold cross-validation.

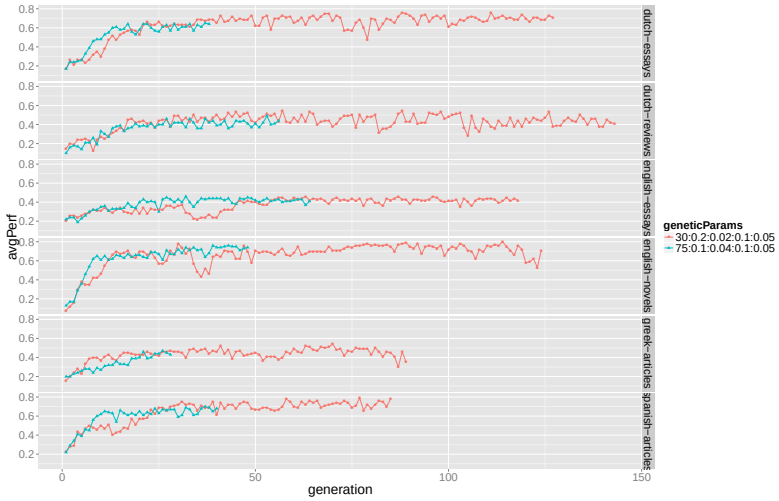

Figure 2. Average performance by generation during the main learning stage. *GeneticParams* contains the following parameters: population, breeders proportion; mutation probability; elitism proportion; random proportion.

5.2 Observations

Below we present some characteristics observed in the configurations which were selected by the genetic learning process (in the case of the fine-grained strategy):

Observation types In most cases only a few observations types are selected (from 3 to 11). Several POS n-grams (as well as POS/tokens combinations) are selected in the five cases where they are available; words n-grams are also selected in most cases, but characters n-grams never are. The word length and the type/token ratio are used in half of the cases. At least one knid of stop words n-grams is selected in 4 datasets.

Consistency, divergence, confidence and distance methods The Bhattacharrya coefficient is used the main criterion for divergence in most cases, and in particular in all datasets where the median number of known documents is higher than 1. The consistency value is actually not used at all in most cases (4): the system choses to use only divergence as confidence to select the relevant observations. The most simple distance metrics are selected in general (mean of the difference, euclidean or cosine distance), but half of the time the frequency is weighted with the confidence score. The number of observations taken into account seems to depend mostly on the dataset.

Learning stage

The decision tree regression (M5P) is selected most of the time for the scoring model. The confidence model stage is selected in only one case; this must be because, in general, the errors made by this classification model are more costly in performance than the benefit of assigning 0.5 scores.

5.3 Earlybird test set and final model selection

Thanks to the *Tira* system [2], we were able to evaluate both strategies on the "earlybird corpus". Figure 3 shows the performance of the two strategies on the training set (by cross-validation) and the earlybird test set; the results obtained on the training set by cross-validation were always better with the fine-grained strategy, but in two cases they were better with the robust strategy on the earlybird test set. This is of course an expected consequence of how the two strategies were defined: the robust strategy is usually not as good as the fine-grained one, but is less data-independent; conversely the fine-grained strategy is more prone to overfitting.

But, more interestingly, we noticed that, with the fine-grained strategy, the performance drops much more (between the training set and the earlybird test set) when the dataset has only a small number of known documents by case in average, especially on the datasets in which most cases contain only one known document. In table 1 we report the performance and compute the drop in performance in all cases, and for each dataset the difference between this value and the average value; based on this difference, we can observe that the drop in performance correlates quite strongly with the (low) number of known documents by case for the fine-grained strategy, whereas it does not at all for the robust strategy.

As a consequence, we decided to use both strategies in our official submission: for the three datasets where there is only one known document (Dutch essays and reviews and English novels), the model corresponding to the robust strategy is used instead of the fine-grained model.

5.4 Results

Table 2 shows the performance obtained on each dataset by both strategies on the training set, the earlybird test set and the final test set, as well as our official ranking.[14] In particular, it shows that our decision to use the robust approach in three cases was good: it performed better than any of the two original strategies taken independently. However our hypothesis that this was linked with the low number of known documents might not hold, since our results on the English novels are quite low compared to the other participants' results, and this would not have happened with the fine-grained strategy. Overall, our system was among the best in this task, ranking third among 13 in average performance.

[14] The results provided at the time of writing have not been made official yet, therefore changes can still happen in the ranking.

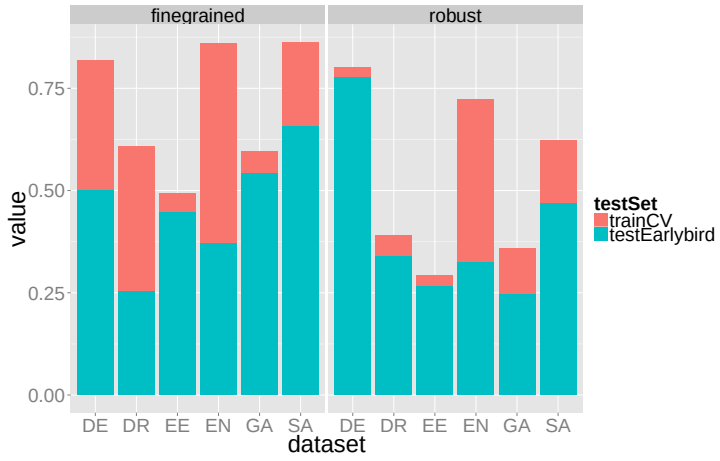

Figure 3. Performance of the fine-grained and robust strategies on the training set and the early-bird test set (the datasets are identified by their initials, their full name can be found in table 1 below).

Table 1. Comparison of the performance on the training set and the earlybird corpus, and impact of the number of known documents by case depending on the strategy. The penultimate column is the difference in performance between the two datasets, and the last column is the difference between the aforementioned value and the average value for the same strategy. Finally the Spearman correlation is calculated between the value in the last column and the mean number of known documents.

Dataset	Known docs/case		Strategy	Perf. training	Perf. Earlybird	Perf. drop	Diff. average
Dutch essays	mean	1.79	robust	0.802	0.777	-0.025	+0.103
	median	1	fine-g.	0.817	0.501	-0.316	-0.071
Dutch reviews	mean	1.02	robust	0.389	0.338	-0.051	+0.077
	median	1	fine-g.	0.608	0.253	-0.355	-0.111
English essays	mean	2.64	robust	0.292	0.265	-0.027	+0.101
	median	3	fine-g.	0.493	0.446	-0.047	+0.198
English novels	mean	1.00	robust	0.722	0.324	-0.398	-0.270
	median	1	fine-g.	0.860	0.370	-0.490	-0.245
Greek articles	mean	2.85	robust	0.359	0.246	-0.113	+0.015
	median	3	fine-g.	0.595	0.541	-0.054	+0.191
Spanish articles	mean	5.00	robust	0.622	0.468	-0.154	-0.026
	median	5	fine-g.	0.863	0.657	-0.206	+0.039

Correlation between Diff. average and mean known docs by case		
	robust	0.77
	fine-g.	0.03

Table 2. Results on all datasets with both strategies. The "mixed" column for the final test set corresponds to our official submission. Remark: there were 13 participants in this task.

Dataset	Training set CV		Earlybird test set			Final test set			
	robust	fine-grained	robust	fine-grained	mixed	robust	fine-grained	mixed	rank
Dutch essays	0.802	0.817	0.777	0.501	0.777	0.755	0.563	0.777	4
Dutch reviews	0.389	0.608	0.338	0.253	0.338	0.375	0.350	0.375	3
English essays	0.292	0.493	0.265	0.446	0.446	0.325	0.372	0.372	3
English novels	0.722	0.860	0.324	0.370	0.324	0.313	0.352	0.313	8
Greek articles	0.359	0.595	0.246	0.541	0.541	0.436	0.565	0.565	3
Spanish articles	0.622	0.863	0.468	0.657	0.657	0.335	0.634	0.634	2
Average	0.531	0.706	0.403	0.461	0.514	0.423	0.473	0.502	3

Acknowledgments

We are grateful to the reviewers for their valuable feedback, and to the organizers of the task for their hard work and their availability.

This research is supported by the Science Foundation Ireland (Grant 12/CE/I2267) as part of the Centre for Global Intelligent Content (www.cngl.ie) funding at Trinity College, University of Dublin.

References

1. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bulletin of Cal. Math. Soc. 35(1), 99–109 (1943)
2. Gollub, T., Potthast, M., Beyer, A., Busse, M., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Recent trends in digital text forensics and its evaluation. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 4th International Conference of the CLEF Initiative (CLEF 13). Springer (September 2013)
3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
4. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to platt's SMO algorithm for SVM classifier design. Neural Comput. 13(3), 637–649 (Mar 2001)
5. Moreau, E., Vogel, C.: Style-based Distance Features for Author Verification - Notebook for PAN at CLEF 2013. In: CLEF 2013 Evaluation Labs and Workshop - Working Notes Papers. p. Online proceedings. Valencia, Spain (Sep 2013)
6. Peñas, A., Rodrigo, A.: A simple measure to assess non-response. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 1415–1424. Association for Computational Linguistics, Portland, Oregon, USA (June 2011), http://www.aclweb.org/anthology/P11-1142
7. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
8. Schmid, H.: Improvements in part-of-speech tagging with an application to german. In: Proceedings of the ACL SIGDAT-Workshop. pp. 47–50 (1995)