A Generic Authorship Verification Scheme Based on Equal Error Rates

Notebook for PAN at CLEF 2015

Oren Halvani and Christian Winter

Fraunhofer Institute for Secure Information Technology SIT
Rheinstrasse 75, 64295 Darmstadt, Germany
{FirstName.LastName}@SIT.Fraunhofer.de

Abstract We present a generic authorship verification scheme for the PAN-2015 identification task. Our scheme uses a two-step training phase on the training corpora. The first phase learns individual feature category parameters as well as decision thresholds based on equal error rates. The second phase builds feature category ensembles which are used for majority vote decisions because ensembles can outperform single feature categories. All feature categories used in our method are very simple to gain multiple advantages: Our method is entirely independent of any external linguistic resources (even word lists), and hence it can easily applied to many languages. Moreover, the classification is very fast due to simple features. Additionally, we make use of parallelization. The evaluation of our scheme on a 40% split (which we did not use for training) of the official PAN-2015 training corpus led to an average corpus accuracy of 68.12%; in detail 60% for the Dutch, 67.5% for the English, 60% for the Greek and 85% for the Spanish subcorpus. The overall computation runtime was approximately 27 seconds.

1 Introduction

Authorship analysis forms a sub-discipline of digital text forensics (a branch of digital forensics). *Authorship attribution* (AA) and *authorship verification* (AV) can be seen as the two major disciplines of authorship analysis [4].

AA deals with the problem of attributing a given anonymous text to exactly one author, given a training set of authors for which text samples of undisputed authorship are available [4]. AV, on the other hand, aims to determine whether a given text was written by a certain author $\mathcal{A}$ [4]. The training set in this case comprises only text samples from the supposed author $\mathcal{A}$, that can be analyzed for distinct features that make up the writing style of $\mathcal{A}$. If it differs significantly from the writing style of the given text, the authorship is considered as false, otherwise true. AVcan be understood as a specialized task of AA with an open set of candidate authors. Conversely, AA can be treated as a sequence of verification problems regarding the given candidate authors [2] and should be solvable if the verification problem is solved [3].

In this work we propose a new intrinsic AV method that offers a number of benefits. The method provides a universal threshold per language and feature category, used to accept or reject the alleged authorship of a document. Here, *universal* refers to the ability to generalize across different genres and topics of the texts. In addition, the model generated by the method is highly flexible and can be extended incrementally in order to handle new languages, genres, topics or features. Besides, the method neither requires complex machine learning algorithms nor error-prone natural language processing techniques nor external linguistic resources. Furthermore, it features a very low computational complexity, compared to other existing approaches (including our previous PAN-approaches).

2 Features

Features are the core of any AV method. Without features it is not possible to model individual writing styles of authors. Many researchers across different fields (linguistics, computer science, mathematics, etc.) have attempted to distinguish features in various ways. As a result, features have been classified into *topic features*, *non-topic features*, *stylistic features*, *lexical features*, *syntactic features*, *semantic features*, etc. In order to avoid using these terms, which can be very misleading, we decided to use the generic term of *features*, which we sub-categorize into so-called feature categories. Regarding our approach, we make use of nine feature categories, which are listed in Table 1.

F_i	**Name**	**Feature description**	**Parameter range**
F_1	Punctuation n-grams	A sequence of n consecutive punctuation marks (commas, hyphens, etc.) taken from $\mathcal{D}$ after reduction to punctuation characters.	$n \in \{1, 2, \ldots, 10\}$
F_2	Character n-grams	A sequence of n consecutive characters in $\mathcal{D}$.	$n \in \{1, 2, \ldots, 10\}$
F_3	$n\%$ frequent tokens	The $n\%$ most frequently occurring tokens in $\mathcal{D}$ (for $n \approx 30$ that will be mostly function words).	$n \in \{5, 10, \ldots, 50\}$
F_4	Token k-prefixes	The first k characters of a token.	$k \in \{1, 2, 3, 4\}$
F_5	Token k-suffixes	The last k characters of a token.	$k \in \{1, 2, 3, 4\}$
F_6	Token k-prefix n-grams	The first k characters of each token within a token n-gram.	$n \in \{2, 3, 4\}$, $k \in \{1, 2, 3, 4\}$
F_7	Token k-suffix n-grams	The last k characters of each token within a token n-gram.	$n \in \{2, 3, 4\}$, $k \in \{1, 2, 3, 4\}$
F_8	n-prefixes– k-suffixes	The first n and last k characters of a token.	$n, k \in \{1, 2, 3, 4\}$
F_9	n-suffixes– k-prefixes	The last n characters of a token and the first k characters of the next token.	$n, k \in \{1, 2, 3, 4\}$

Table 1: The nine feature categories used in our AVscheme.

The extraction of features from a text can be performed on any linguistic layer (e.g., *character layer*, *morphological layer*, *lexical layer*, *syntactic layer* or *semantic layer*). Depending on the layer, different tools are required for the extraction process. For example, extracting features from the *lexical layer*, involves tokenizers, stemmers, or lemmatizers while extracting features from the *semantic layer* requires part-of-speech taggers, parsers, or thesauri [4].

A feature can be *explicit*, i.e extracted directly from the text, or *implicit*, i.e. extracted after the text has been preprocessed. Simple letters, for example, can be seen as explicit features, as they can be looked up in the text itself. In contrast, more complex features like part-of-speech tags (adjectives, articles, pronouns, etc.) are implicit, since they are not annotated in the text and thus can only be extracted after the text has been preprocessed. In our method we focused only on explicit features, extracted from the first four mentioned layers. There are a number of reasons for this decision:

1. A previous observation [1] has shown that implicit features not necessarily lead to promising results in contrast to explicit features (for example character n-grams).
2. Implicit features rely often on natural language processing models that are bounded to only one language and thus are not generalizable.
3. Implicit features often cause a high computation runtime since various types of natural language processing techniques are utilized.

3 Used corpora

In this section we present the training and test corpora used by our method and describe their inner structure.

3.1 Training and test corpora

In order to construct and evaluate our AVmodel, we used the publicly released PAN-2015 training data set[1] denoted by $\mathcal{C}_{\text{Pan15}}$. This corpus is divided into four subcorpora $\mathcal{C}_{\text{Dutch}}$, $\mathcal{C}_{\text{English}}$, $\mathcal{C}_{\text{Greek}}$ and $\mathcal{C}_{\text{Spanish}}$, where each sub corpus consists of 100 problems $\{\rho_1, \rho_2, \ldots, \rho_{100}\}$.

During the time our method was created and trained, the official test corpus has not been released. Hence, we used 60% of each subcorpus as a training and the remaining 40% as a test corpus.

We denote[2] the training corpus as $\mathbb{C}_{\text{Train}} = \{\mathcal{C}_{\text{Tr-Dutch}}, \mathcal{C}_{\text{Tr-English}}, \mathcal{C}_{\text{Tr-Greek}}, \mathcal{C}_{\text{Tr-Spanish}}\}$ and the test corpus as $\mathbb{C}_{\text{Test}} = \{\mathcal{C}_{\text{Te-Dutch}}, \mathcal{C}_{\text{Te-English}}, \mathcal{C}_{\text{Te-Greek}}, \mathcal{C}_{\text{Te-Spanish}}\}$. $\mathbb{C}_{\text{Train}}$ serves exclusively to construct the models (parameters, thresholds and ensembles) while $\mathbb{C}_{\text{Test}}$ is used to test our AVscheme.

[1] The original file name is *pan15-authorship-verification-training-dataset-2015-04-19*.

[2] The prefixes **Tr** and **Te** denote if a corpus $\mathcal{C} \in (\mathbb{C}_{\text{Train}} \cup \mathbb{C}_{\text{Test}})$ is a training or a test corpus.

3.2 Corpus inner structure

The inner structure of the used corpora (whether training or testing) in our method is described as follows. A corpus $\mathcal{C}$ comprises a set of problems $\{\rho_1, \rho_2, \ldots\}$, where a problem $\rho = (\mathcal{D}_{\mathcal{A}_?}, \mathbb{D}_{\mathcal{A}})$ consists of a questioned document $\mathcal{D}_{\mathcal{A}_?}$ and a set of known documents $\mathbb{D}_{\mathcal{A}}$. The number of known documents in each problem ρ varies from one to five documents. The lengths of the texts in these corpora vary between a few hundred and a few thousand words. The distribution of true and false authorships in each corpus is uniform[3]. We denote a true authorship by `Y` and a false authorship by `N`.

4 Our method

In this section we explain first which kind of preprocessing is applied on the training and test corpora. Next, we introduce both models our method relies on, and finally, we describe the treatment of verification problems in the test corpora which involves (among other things) similarity calculation and classification.

4.1 Preprocessing

Given some corpus $\mathcal{C}$, for each document $\mathcal{D}$ within $\mathcal{C}$, we replace newlines, tabs and multiple spaces by only one space. This enables a better extraction of several features, as for instance, character n-grams. Furthermore, we concatenate all known documents within each problem ρ into one document, such that (from now on) each ρ consists only of the two documents $\mathcal{D}_{\mathcal{A}}$ and $\mathcal{D}_{\mathcal{A}_?}$. Besides this, no other preprocessing techniques are applied on the data.

4.2 Model training

Our AVscheme depends on two models that are learned on each training corpus $\mathcal{C} \in \mathbb{C}_{\text{Train}}$ mentioned in Section 3. The first model, denoted as $\mathcal{M}_{\mathbb{F}}$, includes those parameters for each feature category (listed in Table 1) that lad to the highest accuracy on $\mathcal{C}$. Moreover, $\mathcal{M}_{\mathbb{F}}$ includes for each feature category $F_i \in \mathbb{F}$ a threshold θ_{F_i}, which is determined as threshold of the equal error rate (EER). The second model, denoted as $\mathcal{M}_{\mathcal{E}}$, builds on $\mathcal{M}_{\mathbb{F}}$ and includes for each corpus $\mathcal{C}$ an ensemble of feature categories $F_1, F_2, \ldots, F_9$ that lad to the highest accuracy.

Constructing $\mathcal{M}_{\mathbb{F}}$ In order to construct the feature category model $\mathcal{M}_{\mathbb{F}}$, we use the Algorithms $1-6$. Note that some specific methods are not listed, as for instance Length() or LanguageOfCorpus($\mathcal{C}$) as the semantics behind these methods is obvious.

Constructing $\mathcal{M}_{\mathcal{E}}$ In order to construct the ensemble model $\mathcal{M}_{\mathcal{E}}$, we use the Algorithms $7-8$.

[3] Note that the uniform `Y` / `N` distribution is important for the training phase of our method.

```
Input: $\mathbb{F} = \{F_1, F_2, \ldots, F_9\}, \mathbb{C}_{\text{Train}} = \{\mathcal{C}_{\text{Tr-Dutch}}, \mathcal{C}_{\text{Tr-English}}, \mathcal{C}_{\text{Tr-Greek}}, \mathcal{C}_{\text{Tr-Spanish}}\}$
Output: $\mathcal{M}_{\mathbb{F}}$

for $F \in \mathbb{F}$ do
    // Semantic: Language $\rightarrow (n, k), (\theta_F, \mathcal{C}_{Accuracy})$
    $ThresholdDictionary \leftarrow \{\langle\ ,\ \langle(\ ,\ ), (\ ,\ )\rangle\rangle\}$
    for $\mathcal{C} \in \mathbb{C}_{Train}$ do
        $\mathcal{L} \leftarrow$ LanguageOfCorpus($\mathcal{C}$)
        // The first empty tuple denotes the $(n, k)$
           parameters.
        // The second empty tuple denotes $(\theta_F, \mathcal{C}_{Accuracy})$.
        $ThresholdDictionary$.Add($\langle\mathcal{L}, \langle(\ ,\ ), (\ ,\ )\rangle\rangle$)
        $ThresholdLanguageDictionary \leftarrow ThresholdDictionary[\mathcal{L}]$

        for $k \in$ F's k-interval do
            for $n \in$ F's n-interval do
                $\theta_F \leftarrow$ DetermineThreshold($\mathcal{C}, F, n, k$)
                $\mathcal{C}_{Accuracy} \leftarrow$ CalculateCorpusAccuracy($\mathcal{C}, \theta_F, F, n, k$)
                $ThresholdLanguageDictionary$.Add( $\langle(n, k), \langle(\theta_F, \mathcal{C}_{Accuracy})\rangle\rangle$ )
            end
        end
    end

    /* Here we construct the model $\mathcal{M}_{\mathbb{F}}$, which holds for
       each language (represented through $\mathcal{C}$) those $(n, k)$
       parameters and $\theta_F$ that lead to the highest
       accuracy on $\mathcal{C}$.                                    */

    $\mathcal{M}_{\mathbb{F}} \leftarrow \{\langle(), \langle(\ ,\ ,\ ,\ )\rangle\rangle\}$ // Semantic: Language
        $\rightarrow (n, k, \theta_F, \mathcal{C}_{Accuracy})$.

    for $entry \in ThresholdDictionary$ do
        $\mathcal{L} \leftarrow$ LanguageOfCorpus(entry.Key)
        promisingTuple $\leftarrow$ Select the tuple $(n, k, \theta_F, \mathcal{C}_{Accuracy})$ from entry where
        $\mathcal{C}_{Accuracy}$ is the maximum among all tuples within entry.
        $\mathcal{M}_{\mathbb{F}}$.Add(($\mathcal{L}$, promisingTuple))
    end
end
return $\mathcal{M}_{\mathbb{F}}$
```

Algorithm 1: Construct the $\mathcal{M}_{\mathbb{F}}$ model.

```
**Input**: $\mathcal{C}, F, n, k$
**Output**: $\theta_F$

Problem2SimilarityDictionary $\leftarrow$ ComputeProblemsSmilarities($\mathcal{C}, F, n, k$)
psaList $\leftarrow$ Select all tuples $\langle(\rho, \textit{sim}, \textit{trueAuthor})\rangle$ from
Problem2SimilarityDictionary
$Y$ = *psaList*.Filter(*trueAuthor* == "Y").Select(*sim*).OrderByAscending()
$N$ = *psaList*.Filter(*trueAuthor* == "N").Select(*sim*).OrderByAscending()
$\theta_F \leftarrow$ DetermineThresholdViaEER($Y, N$);
**return** $\theta_F$
```

Algorithm 2: DetermineThreshold($\mathcal{C}, F, n, k$).

```
**Input**: $\mathcal{C}, F, n, k$
**Output**: Problem2SimilarityDictionary

*Problem2SimilarityDictionary* = { }
**for** $\rho \in \mathcal{C}$ **do**
    *sim* $\leftarrow$ ComputeProblemSmilarity($\mathcal{D}_{\mathcal{A}_?}, \mathcal{D}_{\mathcal{A}}, F, n, k$)
    *trueAuthor* $\leftarrow$ GetTruth($\rho$)
    *Problem2SimilarityDictionary*.Add($\langle\rho, (F, \textit{sim}, \textit{trueAuthor})\rangle$)
**end**
**return** *Problem2SimilarityDictionary*
```

Algorithm 3: ComputeProblemsSmilarities($\mathcal{C}, F, n, k$).

```
**Input**: $\mathcal{D}_{\mathcal{A}_?}, \mathcal{D}_{\mathcal{A}}, F, n, k$
**Output**: Problem2SimilarityDictionary

$\mathcal{D}_{\mathcal{A}}$-*Features* $\leftarrow$ ExtractFeatures($\mathcal{D}_{\mathcal{A}}, F, n, k$)
$\mathcal{D}_{\mathcal{A}_?}$-*Features* $\leftarrow$ ExtractFeatures($\mathcal{D}_{\mathcal{A}_?}, F, n, k$)
$\mathcal{D}_{\mathcal{A}}$-*NormalizedFeatureDictionary* $\leftarrow$
NormalizeFeaturesByRelativeFrequency($\mathcal{D}_{\mathcal{A}}$-*Features*)
$\mathcal{D}_{\mathcal{A}_?}$-*NormalizedFeatureDictionary* $\leftarrow$
NormalizeFeaturesByRelativeFrequency($\mathcal{D}_{\mathcal{A}_?}$-*Features*)
*Vocabulary* $\leftarrow$ $\mathcal{D}_{\mathcal{A}}$-*NormalizedFeatureDictionary.Keys*
$\cup$ $\mathcal{D}_{\mathcal{A}}$-*NormalizedFeatureDictionary.Keys*
$\mathcal{D}_{\mathcal{A}}$-*VocabularyDictionary* $\leftarrow$
Restrict2Vocabulary($\mathcal{D}_{\mathcal{A}}$-*NormalizedFeatureDictionary*)
$\mathcal{D}_{\mathcal{A}_?}$-*VocabularyDictionary* $\leftarrow$
Restrict2Vocabulary($\mathcal{D}_{\mathcal{A}_?}$-*NormalizedFeatureDictionary*)
$X \leftarrow$ $\mathcal{D}_{\mathcal{A}}$-*VocabularyDictionary.Values*
$Y \leftarrow$ $\mathcal{D}_{\mathcal{A}_?}$-*VocabularyDictionary.Values*
$\mathit{dist} \leftarrow \sum_{i=1}^{m} |X[i] - Y[i]|$ // Manhattan Distance
$\mathit{sim} \leftarrow \frac{1}{1+\mathit{dist}}$ // Linear similarity transformation

**return** *sim*
```

Algorithm 4: ComputeProblemSmilarity($\mathcal{D}_{\mathcal{A}_?}, \mathcal{D}_{\mathcal{A}}, F, n, k$).

```
Input: $\mathcal{C}, F, n, k$
Output: $\theta_F$

if Length(Y) $\neq$ Length(N) then
    Exception("The number of $Y$- does not equal the number of $N$-cases !")
len $\leftarrow$ Length($Y$)
i $\leftarrow 0$
j $\leftarrow$ len $-1$
$\theta_F \leftarrow 0.5$
for $k \leftarrow 0, k < len, k$++ do
    if $Y[i] < N[j]$ then
        i++
        j--
        continue
    if $Y[i] == N[j]$ then
        $\theta_F \leftarrow Y[i]$
        break
    if $Y[i] > N[j]$ then
        if $i == 0$ && $j == len - 1$ then
            $\theta_F \leftarrow (Y[i] + N[j])/2$
        else
            min $\leftarrow$ Min($Y[i], N[j+1]$)
            max $\leftarrow$ Max($Y[i-1], N[j]$)
            $\theta_F \leftarrow$ (min + max)$/2$
        break
end
if i == len then
    $\theta_F = (Y[i-1] + N[j+1])/2$
return $\theta_F$
```

Algorithm 5: DetermineThresholdViaEER(Y, N).

```
**Input**: $\mathcal{C}, \theta_F, F, n, k$
**Output**: $\mathcal{C}_{Accuracy}$

$truePositives \leftarrow 0$
$Problem2SimilarityDictionary \leftarrow \text{ComputeProblemsSmilarities}(\mathcal{C}, F, n, k)$
**for** $\langle \rho, (F, \text{sim}, \text{trueAuthor}) \rangle$ *in Problem2SimilarityDictionary* **do**
    $predictedAuthor \leftarrow$ "Undefined"
    **if** $sim > \theta_F$ **then**
        $predictedAuthor \leftarrow$ "Y"
    **else if** $sim < \theta_F$ **then**
        $predictedAuthor \leftarrow$ "N"
    **if** $predictedAuthor == trueAuthor$ **then**
        $truePositives$++
**end**
$\mathcal{C}_{Accuracy} = \frac{100}{|\mathcal{C}|} \cdot truePositives$
**return** $\mathcal{C}_{\text{Accuracy}}$
```

Algorithm 6: CalculateCorpusAccuracy($\mathcal{C}, \theta_F, F, n, k$).

4.3 Model testing

Now that we have constructed the models we need also to test them. This is done in Algorithm 9.

5 Evaluation

In this section we describe the evaluation results. First, we present the models $\mathcal{M}_{\mathbb{F}}$ and $\mathcal{M}_{\mathcal{E}}$ learned from the training corpora $\mathbb{C}_{\text{Train}} = \{\mathcal{C}_{\text{Tr-Dutch}}, \mathcal{C}_{\text{Tr-English}}, \mathcal{C}_{\text{Tr-Greek}}, \mathcal{C}_{\text{Tr-Spanish}}\}$. Afterwards, we show (given both pre-trained models) the evaluation results on the test corpora $\mathbb{C}_{\text{Test}} = \{\mathcal{C}_{\text{Te-Dutch}}, \mathcal{C}_{\text{Te-English}}, \mathcal{C}_{\text{Te-Greek}}, \mathcal{C}_{\text{Te-Spanish}}\}$.

5.1 Determining $\mathcal{M}_{\mathbb{F}}$

The training on $\mathbb{C}_{\text{Train}}$ led to the model $\mathcal{M}_{\mathbb{F}} = \{\mathcal{M}_{F_1}, \mathcal{M}_{F_2}, \ldots, \mathcal{M}_{F_9}\}$ which is given in the Tables 2$-$10.

5.2 Determining $\mathcal{M}_{\mathcal{E}}$

Given $\mathcal{M}_{\mathbb{F}}, \mathbb{C}_{\text{Train}}$ and $\mathbb{F}$ as an input, we applied Construct$\mathcal{M}_{\mathcal{E}}(\mathcal{M}_{\mathbb{F}}, \mathbb{C}_{\text{Train}}, \mathbb{F})$ on all training subcorpora. The resulting model $\mathcal{M}_{\mathcal{E}}$ is given in Table 11.

```
Input: $\mathcal{M}_{\mathbb{F}}, \mathbb{C}_{\text{Train}}, \mathbb{F}$
Output: $\mathcal{M}_{\mathcal{E}}$

$\mathcal{M}_{\mathcal{E}} \leftarrow \{\ \}$
// SuitableEnsembles includes only odd-numbered
   ensembles due to a majority voting.
SuitableEnsembles = GetFeatureCategoriesPowerSet($\mathbb{F}$)
for $\mathcal{C} \in \mathbb{C}_{Train}$ do
    $\mathbb{F}_{pairings} \leftarrow$ Construct$\mathbb{F}_{pairings}(\mathcal{M}_{\mathbb{F}}, \mathcal{C}, \mathbb{F})$
    // Semantic: ensemble $\rightarrow$ corpusAccuracy
    featureCategoryComboAccuracy$\leftarrow \{\langle(), ()\rangle\}$
    for ensemble $\in$ SuitableEnsembles do
        predictions $\leftarrow$ CreateFeatureCategoryEnsembleResults(ensemble, $\mathcal{C}$)
        corpusAccuracy $\leftarrow$ EvaluatePredictionsViaTruthFile(predictions, $\mathcal{C}$)
        featureCategoryComboAccuracy.Add((ensemble, corpusAccuracy))
    end
    featureCategoryComboAccuracy $\leftarrow$ Sort all entries (ensemble,
    corpusAccuracy) $\in$ featureCategoryComboAccuracy by descending,
    according to corpusAccuracy.
    $\mathcal{E} \leftarrow$ Select the most promising ensemble from
    featureCategoryComboAccuracy, which led to the highest corpusAccuracy
    $\mathcal{M}_{\mathcal{E}}$.Add(bestEnsemble)
end
return $\mathcal{M}_{\mathcal{E}}$
```

Algorithm 7: Construct$\mathcal{M}_{\mathcal{E}}(\mathcal{M}_{\mathbb{F}}, \mathbb{C}_{\text{Train}}, \mathbb{F})$.

```
Input: $\mathcal{M}_{\mathbb{F}}, \mathcal{C}, \mathbb{F}$
Output: $\mathbb{F}_{pairings}$

// Semantics: $F \rightarrow \{\langle \rho, (\theta_F, sim) \rangle\}$.
$\mathbb{F}_{pairings} \leftarrow \{\langle (), (,) \rangle\}$
for $F \in \mathbb{F}$ do
    $F_{pairings} \leftarrow \{\langle (), (,) \rangle\}$
    $\mathbb{F}_{pairings}$.Add($\langle F, F_{pairings} \rangle$)
    for $\rho \in \mathcal{C}$ do
        $\mathcal{M}_F \leftarrow$ LoadModelFrom$\mathcal{M}_{\mathbb{F}}(F)$
        $n \leftarrow$ LoadNFrom$\mathcal{M}_F()$
        $k \leftarrow$ LoadKFrom$\mathcal{M}_F()$
        $\theta_F \leftarrow$ LoadThresholdFrom$\mathcal{M}_F()$
        $sim \leftarrow$ ComputeProblemSmilarity($\mathcal{C}, F, n, k$)
        $F_{pairings}$.Add($\langle \rho, (\theta_F, sim) \rangle$)
    end
end

return $\mathbb{F}_{\text{pairings}}$
```

Algorithm 8: Construct$\mathbb{F}_{pairings}(\mathcal{M}_{\mathbb{F}}, \mathcal{C}, \mathbb{F})$.

```
Input: $\mathcal{M}_{\mathbb{F}}, \mathcal{M}_{\mathcal{E}}, \mathbb{C}_{\text{Test}}, \mathbb{F}$
Output:

for $\mathcal{C} \in \mathbb{C}_{Test}$ do
    $\mathbb{F}_{pairings} \leftarrow$ Construct$\mathbb{F}_{pairings}(\mathcal{M}_{\mathbb{F}}, \mathcal{C}, \mathbb{F})$
    $\mathcal{L} \leftarrow$ LanguageOfCorpus($\mathcal{C}$)
    $\mathcal{E} \leftarrow$ Select the most promising ensemble from $\mathcal{M}_{\mathcal{E}}$, for $\mathcal{C}$ with respect to $\mathcal{L}$
    predictions $\leftarrow$ CreateFeatureCategoryEnsembleResults($\mathcal{E}$, $\mathcal{C}$)
    $\mathcal{C}_{Accuracy} \leftarrow$ EvaluatePredictionsViaTruthFile(predictions, $\mathcal{C}$)
    Print(predictions, $\mathcal{C}_{Accuracy}$)
end
```

Algorithm 9: Apply the AVscheme, based on the pre-trained models, on test corpora $\mathbb{C}_{\text{Test}}$.

$\mathcal{C}$	n	θ_{F_1}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	6	0,34	63.33 %
$\mathcal{C}_{\text{Tr-English}}$	1	0,66	76.67 %
$\mathcal{C}_{\text{Tr-Greek}}$	2	0,64	76.67 %
$\mathcal{C}_{\text{Tr-Spanish}}$	1	0,78	83.33 %

Table 2: The $\mathcal{M}_{F_1}$ model.

$\mathcal{C}$	n	θ_{F_2}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	1	0,81	70 %
$\mathcal{C}_{\text{Tr-English}}$	1	0,79	70 %
$\mathcal{C}_{\text{Tr-Greek}}$	7	0,34	80 %
$\mathcal{C}_{\text{Tr-Spanish}}$	1	0,89	70 %

Table 3: The $\mathcal{M}_{F_2}$ model.

$\mathcal{C}$	n	θ_{F_3}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	10	0,46	66.67 %
$\mathcal{C}_{\text{Tr-English}}$	5	0,43	56.67 %
$\mathcal{C}_{\text{Tr-Greek}}$	45	0,45	76.67 %
$\mathcal{C}_{\text{Tr-Spanish}}$	10	0,57	70 %

Table 4: The $\mathcal{M}_{F_3}$ model.

$\mathcal{C}$	k	θ_{F_4}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	1	0,65	66.67 %
$\mathcal{C}_{\text{Tr-English}}$	4	0,37	63.33 %
$\mathcal{C}_{\text{Tr-Greek}}$	2	0,56	70 %
$\mathcal{C}_{\text{Tr-Spanish}}$	1	0,79	83.33 %

Table 5: The $\mathcal{M}_{F_4}$ model.

$\mathcal{C}$	k	θ_{F_5}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	3	0,44	63.33 %
$\mathcal{C}_{\text{Tr-English}}$	2	0,51	60 %
$\mathcal{C}_{\text{Tr-Greek}}$	1	0,77	76.67 %
$\mathcal{C}_{\text{Tr-Spanish}}$	3	0,51	66.67 %

Table 6: The $\mathcal{M}_{F_5}$ model.

$\mathcal{C}$	n	k	θ_{F_6}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	1	2	0,40	63.33 %
$\mathcal{C}_{\text{Tr-English}}$	3	2	0,34	56.67 %
$\mathcal{C}_{\text{Tr-Greek}}$	4	2	0,33	70 %
$\mathcal{C}_{\text{Tr-Spanish}}$	4	2	0,33	71.67 %

Table 7: The $\mathcal{M}_{F_6}$ model.

$\mathcal{C}$	n	k	θ_{F_7}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	1	2	0,50	66.67 %
$\mathcal{C}_{\text{Tr-English}}$	2	2	0,35	66.67 %
$\mathcal{C}_{\text{Tr-Greek}}$	1	2	0,58	80 %
$\mathcal{C}_{\text{Tr-Spanish}}$	4	3	0,33	66.67 %

Table 8: The $\mathcal{M}_{F_7}$ model.

$\mathcal{C}$	n	k	θ_{F_8}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	2	3	0,34	73.33 %
$\mathcal{C}_{\text{Tr-English}}$	3	2	0,34	66.67 %
$\mathcal{C}_{\text{Tr-Greek}}$	2	1	0,43	76.67 %
$\mathcal{C}_{\text{Tr-Spanish}}$	1	1	0,62	80 %

Table 9: The $\mathcal{M}_{F_8}$ model.

Table 10: The $\mathcal{M}_{F_9}$ model.

$\mathcal{C}$	n	k	θ_{F_9}	$\mathcal{C}_{\text{Accuracy}}$
$\mathcal{C}_{\text{Tr-Dutch}}$	1	1	0,45	70 %
$\mathcal{C}_{\text{Tr-English}}$	3	1	0,35	60 %
$\mathcal{C}_{\text{Tr-Greek}}$	4	1	0,35	76.67 %
$\mathcal{C}_{\text{Tr-Spanish}}$	1	1	0,60	80 %

Table 11: The $\mathcal{M}_{\mathcal{E}}$ model for $\mathbb{C}_{\text{Train}}$.

$\mathcal{C}$	$\mathcal{E}$ **(Ensemble)**
$\mathcal{C}_{\text{Tr-Dutch}}$	$\{F_1, F_8, F_9\}$
$\mathcal{C}_{\text{Tr-English}}$	$\{F_1\}$
$\mathcal{C}_{\text{Tr-Greek}}$	$\{F_1, F_3, F_4, F_6, F_7\}$
$\mathcal{C}_{\text{Tr-Spanish}}$	$\{F_1, F_4, F_6\}$

5.3 Evaluation of the models

The evaluation of our approach (based on the pre-trained models $\mathcal{M}_{\mathbb{F}}$ and $\mathcal{M}_{\mathcal{E}}$) on the test corpus $\mathbb{C}_{\text{Test}}$ led to the results, listed in Table 12. The runtime needed for the entire evaluation was 27 seconds.

Table 12: Results after applying the $\mathcal{M}_{\mathbb{F}}$ and $\mathcal{M}_{\mathcal{E}}$ models on $\mathbb{C}_{\text{Test}}$.

$\mathcal{C}$	**Accuracy**
$\mathcal{C}_{\text{Te-Dutch}}$	60 %
$\mathcal{C}_{\text{Te-English}}$	67.5 %
$\mathcal{C}_{\text{Te-Greek}}$	60 %
$\mathcal{C}_{\text{Te-Spanish}}$	85 %
$\mathbb{C}_{\text{Test}}$	$\varnothing =$ **68.12 %**

6 Conclusion and future work

We presented a generic and fast authorship verification scheme for the Author Identification task within the PAN-2015 competition. Our method provides a number of benefits. The most important benefit (in comparison to our previous PAN-approaches) is that our method finally makes use of trainable models. These models not only contribute to the low runtime since thresholds do not need to be computed over and over again, but also can be extended incrementally and so enable a better generalization in terms of languages, genres or topics. Another benefit that directly comes from the (transparent) models is that it makes the approach better interpretable as one is able to inspect the effects of the different feature category parameters as well as the decision thresholds. In future work we will focus on this detail. A further benefit of our method is that there is no need for deep linguistic processing (e.g., POS-tagging, chunking or parsing) or linguistic resources such as word lists, word-nets, thesauruses, etc. This causes not only a low runtime but also makes the method portable. Besides this, the method can be modified easily regarding its underlying components. For example, the distance function can be replaced by an ensemble of distance functions with almost no effort.

However, there are also many open issues in our approach that leave room for improvement. Here, the most important issue is that at the moment our method is difficult to comprehend, which makes it complicated to understand and implemented by others. Therefore, it is planed in future work to re-factor the algorithms in order to make the scheme more compact.

Acknowledgments. This work was supported by the Center for Advanced Security Research Darmstadt CASED (`www.CASED.de`), funded by the German state government of Hesse under the LOEWE program and by the German Federal Ministry of Education and Research (BMBF) in the funded projects ALPhA and EWV.

References

1. Juola, P., Stamatatos, E.: Overview of the Author Identification Task at PAN 2013. In: Forner, P., Navigli, R., Tufis, D., Ferro, N. (eds.) Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013. CEUR Workshop Proceedings, vol. 1179. CEUR-WS.org (2013), `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-JuolaEt2013.pdf`
2. Koppel, M., Schler, J., Argamon, S., Winter, Y.: The "Fundamental Problem" of Authorship Attribution. English Studies 93(3), 284–291 (2012), `http://dx.doi.org/10.1080/0013838X.2012.668794`
3. Koppel, M., Winter, Y.: Determining if Two Documents are by the Same Author. JASIST 65(1), 178–187 (2014)
4. Stamatatos, E.: A Survey of Modern Authorship Attribution Methods. J. Am. Soc. Inf. Sci. Technol. 60(3), 538–556 (Mar 2009)