VEBAV - A Simple, Scalable and Fast Authorship Verification Scheme

Notebook for PAN at CLEF 2014

Oren Halvani and Martin Steinebach

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

Abstract We present VEBAV - a simple, scalable and fast authorship verification scheme for the Author Identification (AI) task within the PAN-2014 competition. VEBAV (VEctor-Based Authorship Verifier), which is a modification of our existing PAN-2013 approach, is an intrinsic one-class-verification method, based on a simple distance function. VEBAV provides a number of benefits as for instance the independence of linguistic resources and tools like ontologies, thesauruses, language models, dictionaries, spellcheckers, etc. Another benefit is the low runtime of the method, due to the fact that deep linguistic processing techniques like POS-tagging, chunking or parsing are not taken into account. A further benefit of VEBAV is the ability to handle more as only one language. More concretely, it can be applied on documents written in Indo-European languages such as Dutch, English, Greek or Spanish. Regarding its configuration VEBAV can be extended or modified easily by replacing its underlying components. These include, for instance, the classification function, the distance function, the acceptance criterion, the underlying features (including their parameters) and many more. In our experiments we achieved regarding a 20%-split of the PAN 2014 AI-training-corpus an overall accuracy score of 65,83% (in detail: 80% for Dutch-Essays, 55% for Dutch-Reviews, 55% for English-Essays, 80% English-Novels, 70% for Greek-Articles and 55% for Spanish-Articles).

1 Introduction

Authorship Verification (AV) is a sub-discipline of Authorship Analysis [3, Page: 3], which itself is an integral part of digital text forensics. It can be applied in many forensic scenarios as for instance checking the authenticity of written contracts, threats, insults, testaments, etc. where the goal of AV remains always the same: Verify if two documents $\mathcal{D}_{\mathcal{A}}$ and $\mathcal{D}_{\mathcal{A}?}$ are written by the same author $\mathcal{A}$, or not. An alternative reformulation of the goal is to verify the authorship of $\mathcal{D}_{\mathcal{A}?}$, given a set of sample documents of $\mathcal{A}$. Both formulations hold for the AI-task within the PAN-2014 competition. In order to perform AV at least four components are mandatorily required:

- The document $\mathcal{D}_{\mathcal{A}?}$, which should be verified regarding its alleged authorship.
- A training set $\mathbb{D}_{\mathcal{A}} = \{\mathcal{D}_{1\mathcal{A}}, \mathcal{D}_{2\mathcal{A}}, ...\}$, where each $\mathcal{D}_{i\mathcal{A}}$ represents a sample document of $\mathcal{A}$.

- A set of features $\mathbb{F} = \{f_1, f_2, ...\}$, where each f_j (*style marker*) should help to model the writing style of $\mathcal{D}_{\mathcal{A}?}$ and each $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$.
- At least one classification method, which accepts or rejects the given authorship based on $\mathbb{F}$ and a predefined or dynamically determined threshold θ.

From a Machine Learning perspective AV clearly forms an one-class-classification problem [2], due to the fact that $\mathcal{A}$ is the only target class to be distinguished among all other possible classes (authors), where their number can be theoretically infinite.

The aim of this paper is to provide a simple, scalable and fast AV scheme AV scheme, which offers many benefits as for instance promising detection rates, easy implementation, low runtime, independence of language or linguistic resources as well as easy modifiability and expandability. Our proposed AV scheme `VEBAV` is based on our earlier approach regarding the PAN 2013 AI-task, which itself formed a modification of the Nearest Neighbor (NN) one-class classification technique, described by Tax in [4, Page: 69].

In a nutshell, `VEBAV` takes as an input a set of sample documents of a known author ($\mathbb{D}_{\mathcal{A}}$) and exactly one document of an unknown author ($\mathcal{D}_{\mathcal{A}?}$). All documents in $\mathbb{D}_{\mathcal{A}}$ are then concatenated into a big document which, depending on the length, is then splitted again into three or five eqal-sized chunks (such that $\mathbb{D}_{\mathcal{A}}$ is updated by these chunks). Next, feature vectors are constructed from each $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ and from $\mathcal{D}_{\mathcal{A}?}$. Then, a representative is selected among the training feature vectors, which is important to determine the decision regarding the alleged authorship. In the next step distances are calculated between the representative and the remaining training feature vectors, but also between it and the test feature vector. Depending on all calculated distances a twofold decision regarding the alleged authorship is generated, which includes a ternary decision $\delta \in \{$*Yes*, *No*, *Unanswered*$\}$ and a probability score ρ that describes the soundness of δ.

The rest of this paper is structured as follows. In the following section 2 we present all feature sets, which have been used in this paper. After that we describe in section our verification scheme. In 4 we present the corpus that was used to evaluate our method. In section 5 we provide the results regarding the test set, which are based on four experiments. Finally, we draw our conclusions in section 6 and provide some ideas for future work.

2 Features

Style markers (features) are the core of AV, since they are able to approximate writing styles and thus, can help to judge if two texts are originate from the same style. If this holds, it is an indicator that both texts have been written by the same author. In the next subsections we explain from where exactly quantifiable features can be retrieved, which tools are required for the extraction process and finally what kind of feature sets we used in our approach.

2.1 Linguistic layers

Typically, features are extracted from linguistic layers, which can be understood as abstract units within a text. In general, the most important linguistic layers are the following:

- **Phoneme layer:** This layer includes phoneme-based features as for example vowels, consonants or also the more sophisticated supra-segmental or prosodic features. Such features can typically be won out of texts by using (pronouncing) dictionaries (e.g. IPA).
- **Character layer:** This layer includes character-based features as for instance prefixes, suffixes or letter n-Grams, which typically are extracted from texts via regular expressions.
- **Lexical layer:** This layer includes token-based features as for instance function words or POS-Tags (Part-Of-Speech Tags). These features can be extracted from texts via tokenisers (which often are based on simple regular expressions).
- **Syntactic layer:** This layer includes syntax-based features as for instance constituents (e.g. nominal phrases) or collocations. Such features can be extracted by sophisticated regular expressions or by natural language processing tools (e.g. POS-Tagger). However, the latter one is normally bounded to a specific language model and thus, cannot scale to multiple languages. Besides this the runtime of natural language processing tools is much more higher as the runtime caused by regular expressions.
- **Semantic Layer:** This layer includes semantic-based features, e.g. semantic relations (hyponymous, synonymys, meronyms, etc.). Such features require deep linguistic processing, which often rely on external knowledge resources (e.g. WordNet) or complex tools (e.g. parsers, named entity eecognizers, etc.).

In `VEBAV` we only make use of the *Character*, *Lexical* and *Syntactic* linguistic layers, due to their effectiveness and their low runtime.

2.2 Feature sets

In this paper we use the term *feature set* denoted by F, which consists of features belonging to at least one linguistic layer. Table 1 shows 14 feature sets that have been used in our experiments.

2.3 Parameters

The feature sets mentioned in Table 1 can be parameterized in several ways, e.g. by the n-Gram sizes, the number of k for prefixes/suffixes and the number of the top frequent occurring features. It should be emphasised that adjusting such settings can influence the results massively.

F_i	Feature set	Description	Examples
F_1	*Characters*	All kind of characters	{a,b,1,8,#,<,%,!,...}
F_2	*Letters*	All kind of letters	{a,b,α,β,ä,ß,ó,á,ñ,...}
F_3	*Punctuation marks*	Symbols to structure sentences	{.,:,;,-,",(,),...}
F_4	*Word k Prefixes*	The k starting letters of words	example $\rightsquigarrow$ {e,ex,exa,exam,...}
F_5	*Word k Suffixes*	The k ending letters of words	example $\rightsquigarrow$ {e,le,ple,mple,...}
F_6	*Character n-Grams*	Overlapping character-fragments	ex-ample $\rightsquigarrow$ {ex-,x-a,-am,...}
F_7	*Letter n-Grams*	Overlapping letter-fragments	example $\rightsquigarrow$ {exa,xam,amp,...}
F_8	*Tokens*	Segmented character-based units	A [simple] text! $\rightsquigarrow$ {A,[simple],text!}
F_9	*Words*	Segmented letter-based units	A [simple] text! $\rightsquigarrow$ {A,simple,text}
F_{10}	*Token n-Grams*	Overlapping token-fragments	A [simple] text! $\rightsquigarrow$ {A [simple],[simple] text!}
F_{11}	*Word n-Grams*	Overlapping word-fragments	A [simple] text! $\rightsquigarrow$ {A simple,simple text}
F_{12}	*Mix*$_1$	A mix of three feature sets	$F_1 \cup F_3 \cup F_6$
F_{13}	*Mix*$_2$	A mix of three feature sets	$F_1 \cup F_2 \cup F_5$
F_{14}	*Mix*$_3$	A mix of four feature sets	$F_3 \cup F_4 \cup F_5 \cup F_8$

Table 1: All feature sets used in our approach.

3 Proposed Verification Scheme

In this section we give a detailed description of VEBAV. For overview reasons we divided the entire workflow of the algorithm into six subsections, where we first explain what kind of preprocessing we perform on the data. The other five subsections focus on the algorithm itself.

3.1 Preprocessing

In contrast to our approach in PAN-2013 we decided in our current approach neither to apply normalization nor noise reduction techniques. Instead, we treat each text as it is, which turned out to be not only less burdensome but also promising. Our only *preprocessing* is restricted to concatenate all $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ into a single document $\mathcal{D}_{\mathcal{A}}$, which is then splitted into ℓ (near) equal-sized chunks $\mathcal{D}_{1\mathcal{A}}, \mathcal{D}_{2\mathcal{A}}, ..., \mathcal{D}_{\ell\mathcal{A}}$. Here, ℓ is statically set to five chunks if, and only if, the length of $\mathcal{D}_{\mathcal{A}}$ is above 15,000 characters. Otherwise ℓ is statically set to three chunks.

3.2 Vocabulary Generation

In order the form a basis for the construction of feature vectors, we need the build a global vocabulary $\mathcal{V}$. For this, we first generate for $\mathcal{D}_{\mathcal{A}_?}$ and each chunk $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ their corresponding vocabularies $\mathcal{V}_{\mathcal{D}_{\mathcal{A}_?}}$ and $\mathcal{V}_{\mathcal{D}_{1\mathcal{A}}}, \mathcal{V}_{\mathcal{D}_{2\mathcal{A}}}, ..., \mathcal{V}_{\mathcal{D}_{\ell\mathcal{A}}}$. Next, we apply an intersection among all vocabularies to build the global vocabulary:

$$\mathcal{V} = \mathcal{V}_{\mathcal{D}_{\mathcal{A}_?}} \cap \mathcal{V}_{\mathcal{D}_{1\mathcal{A}}} \cap \mathcal{V}_{\mathcal{D}_{2\mathcal{A}}} \cap \ldots \cap \mathcal{V}_{\mathcal{D}_{\ell\mathcal{A}}}$$

3.3 Constructing feature vectors

After $\mathcal{V}$ is generated, the next step is to construct the feature vectors $\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}$ from each $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ and $\mathcal{F}_{\mathcal{A}?}$ from $\mathcal{D}_{\mathcal{A}_?}$. Beforehand, at least one appropriate feature set F_i must be chosen, where *appropriate* refers to:

– Features that appear in all generated vocabularies (such that $\mathcal{V} \neq \emptyset$).
– Features that are able to model high similarity between $\mathcal{D}_{\mathcal{A}_?}$ and $\mathcal{D}_{\mathcal{A}}$ (for the case $\mathcal{A}_? = \mathcal{A}$).
– Features that are able to highly discriminate between the writing style of $\mathcal{A}$ and all other possible authors (for the case $\mathcal{A}_? \neq \mathcal{A}$).

Each feature vector consists of exactly $n = |\mathcal{V}|$ numerical values, where each value represents a relative frequency of a feature within its underlying document, according to chosen F_i.

3.4 Representative Selection

After constructing all feature vectors, a representative (training-) feature vector $\mathcal{F}_{rep}$ must be selected. This step is essential for the later determination of the decision regarding the alleged authorship. In general, VEBAV offers two options to select $\mathcal{F}_{rep}$:

– Selecting $\mathcal{F}_{rep}$ manually (static).
– Selecting $\mathcal{F}_{rep}$ dynamically by using a similarity function (e.g. cosine similarity) between all training feature vectors. Here, $\mathcal{F}_{rep}$ is selected according to the feature vector, who is mostly dissimilar from the others. In other terms $\mathcal{F}_{rep}$ can be understand as an outlier.

3.5 Distances Calculations

In this step we calculate the distances that are needed to determine the decision regarding the alleged authorship. Concretely, we calculate the distances $d_1, d_2, ..., d_\ell$ between $\mathcal{F}_{rep}$ and each $\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}$ as also the distance $d_?$ between $\mathcal{F}_{rep}$ and $\mathcal{F}_{\mathcal{A}?}$. The calculation of these distances requires a predefined distance function dist(X, Y), where X refers here to $\mathcal{F}_{rep}$ and Y to some other feature vector. We integrated a broad range of distance functions into VEBAV, where the majority have been taken from [1]. However, in our experiments we did only make use of the two distance functions *Minkowski* and *Canberra*, due to their promising results over the other distance functions. As a last step we calculate the average regarding the distances of the training feature vectors $d_\varnothing = \frac{1}{\ell}(d_1 + d_2 + \ldots + d_\ell)$.

3.6 Decision Determination

The goal of this step is to construct a twofold decision regarding the alleged authorship, which consists of a ternary decision $\delta \in \{$*Yes*, *No*, *Unanswered*$\}$ and a probability score ρ. In order to calculate these terms both values are required $d_?$ and $d_\varnothing$. For the latter one we use the following adjusted form: $d' = d_\varnothing + (\omega \cdot \tau)$. Here, ω represents a weight and τ a tolerance parameter, which is calculated from the a standard deviation of $d_1, d_2, ..., d_\ell$. The intention behind ω and τ is to handle noisy writing styles, which still might be coined from the same author. With this we calculate ρ and δ as follows:

$$\rho = \frac{1}{1 + \frac{d_?}{d'}}\ , \qquad \delta = \begin{cases} \textit{Yes}, & d_? < d' \\ \textit{No}, & d_? > d' \\ \textit{Unanswered}, & (d_? = d') \wedge (|d_? - d'| < \varepsilon) \end{cases}$$

The semantic behind δ is:

- *Yes*: VEBAV predicts the alleged author as the true author ($\mathcal{A}_? = \mathcal{A}$).
- *No*: VEBAV predicts the alleged author not as the true author ($\mathcal{A}_? \neq \mathcal{A}$).
- *Unanswered*: VEBAV was unable to generate the prediction because $d_?$ and d' are near-equal or due to another unexpected result. Depending on how ε was chosen the number of *Unanswered* decisions can vary considerably. In our experiments we to chose $\varepsilon = 0,001$ as this restricts *Unanswered* decisions when ρ is near 0.5.

4 Used Corpora

Regarding our experiments we used the $\mathcal{C}_{PAN\text{-}14}$ (*PAN-2014 Author Identification*) training corpus, released by the PAN organizers at 22.04.2014. $\mathcal{C}_{PAN\text{-}14}$ consists of 695 problems (in total 2,382 documents), equally distributed regarding true/false authorships. A problem p_i forms a tuple $(\mathbb{D}_{\mathcal{A}_i}, \mathcal{D}_{\mathcal{A}_i?})$, where $\mathbb{D}_{\mathcal{A}_i}$ denotes the training set author $\mathcal{A}_i$ and $\mathcal{D}_{\mathcal{A}_i?}$ the questioned document, which was (or not) written by $\mathcal{A}_i$. Each problem belongs to one of four languages (*Greek*, *Spanish*, *Greek* and *Spanish* and to one of four genres (*Essays*, *Reviews*, *Novels* and *Articles*). For simplification reasons, $\mathcal{C}_{PAN\text{-}14}$ is divided into six subcorpora and thus, can be formulated as $\mathcal{C}_{PAN\text{-}14} = \{\mathcal{C}_{DE}, \mathcal{C}_{DR}, \mathcal{C}_{EE}, \mathcal{C}_{EN}, \mathcal{C}_{GR}, \mathcal{C}_{SP}\}$. This makes it easier to treat each subcorpus independently (e.g. in terms of parameterisations). The full name of each $\mathcal{C} \in \mathcal{C}_{PAN\text{-}14}$ is given below:

- $\mathcal{C}_{DE}$: *Dutch-Essays*
- $\mathcal{C}_{DR}$: *Dutch-Reviews*
- $\mathcal{C}_{EE}$: *English-Essays*
- $\mathcal{C}_{EN}$: *English-Novels*
- $\mathcal{C}_{GR}$: *Greek-Articles*
- $\mathcal{C}_{SP}$: *Spanish-Articles*

For our experiments we used 80% of $\mathcal{C}_{PAN\text{-}14}$ for training and parameter learning (denoted by $\mathcal{C}_{PAN\text{-}Train}$), while the remaining 20% was used for testing (denoted by $\mathcal{C}_{PAN\text{-}Test}$).

5 Evaluation

In this section we carry out our evaluation regarding the $\mathcal{C}_{PAN\text{-}14} = \mathcal{C}_{PAN\text{-}Train} \cup \mathcal{C}_{PAN\text{-}Test}$ corpus. We first explain which performance measures were used and secondly, how the most important parameters were learned from $\mathcal{C}_{PAN\text{-}Train}$. Finally, we evaluate our approach on $\mathcal{C}_{PAN\text{-}Test}$.

5.1 Performance measures

In order to evaluate our approach, we used several performance measures, sharing the following variables:

- $n =$ *Number of problems in* $\mathcal{C}$
- $n_c =$ *Number of correct answers per* $\mathcal{C}$

– n_u = *Number of unanswered problems answers per* $\mathcal{C}$

The first performance measure is:

$$\textit{Accuracy} = \frac{\textit{Number of correct answers per dataset } \mathcal{C}}{\textit{Total number of documents per dataset } \mathcal{C}}$$

Regarding the second performance measure we first define both terms *AUC* and $c@1$ as follows:

$$\textit{AUC} = \sum_{i=1}^{n} \rho \,, \qquad c@1 = \frac{1}{n}\left(n_c + \left(n_u \cdot \left(\frac{n_c}{n}\right)\right)\right)$$

With these two we define the second performance measure: $\textit{AUC} \cdot c@1$. Note, regarding the process of learning the most important parameters on $\mathcal{C}_{\textit{PAN-Train}}$ we only make use of the *Accuracy* measure.

5.2 Experiment I: Finding an optimal λ for the *Minkowski* distance function

The intention behind this experiment was to find an optimal λ parameter, used by the *Minkowski* distance function. Since λ has a strong influence on the classification result it must be well-chosen, in order to generalize across the range of all involved corpora and all feature sets. To achieve this generalization we merged all training subcorpora, such that $\mathcal{C}_{\textit{PAN-Train}} = \mathcal{C}_{DE} \cap \mathcal{C}_{DR} \cap \mathcal{C}_{EE} \cap \mathcal{C}_{EN} \cap \mathcal{C}_{GR} \cap \mathcal{C}_{SP}$ holds. Afterwards, we applied VEBAV on $\mathcal{C}_{\textit{PAN-Train}}$, where as an input we used all mentioned feature sets in Table 1 and the following 14 predefined λ values $\{0.2, 0.4, 0.6, 0.8, 1, 2, ..., 10\}$. We constructed from the results a table, where the rows represent the feature sets $F_1, F_2, \ldots, F_{14}$, while the columns represent the 14 λ values. Next, we derived a row from this table that includes the medians regarding all columns. This row is illustrated in Figure 1.

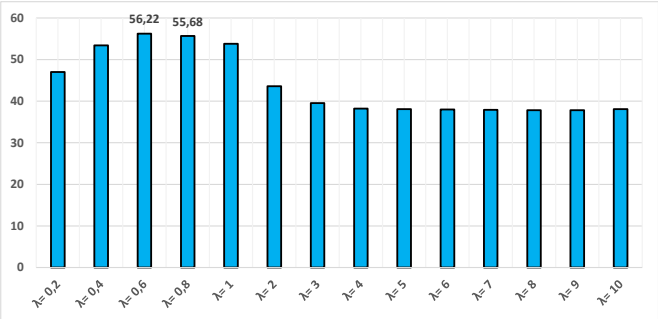

Figure 1. Comparison of different λ values for the *Minkowski* distance function.

As can be seen, an optimal range for for λ is [0.2; 1], where 0.6 seems to be the most promising one in terms of robustness, among all involved feature sets. As a consequence of this analysis, we decided to use $\lambda = 0.6$ for the further experiments.

5.3 Experiment II: Determinig the classification stregth of all feature sets

In this experiment we wanted to compare the classification stregth of all involved feature sets. Hence, we applied VEBAV again on $F_1, F_2, \ldots, F_{14}$, where this time we used only 0.6 as a fixed λ value. The results are illustrated in Figure 2.

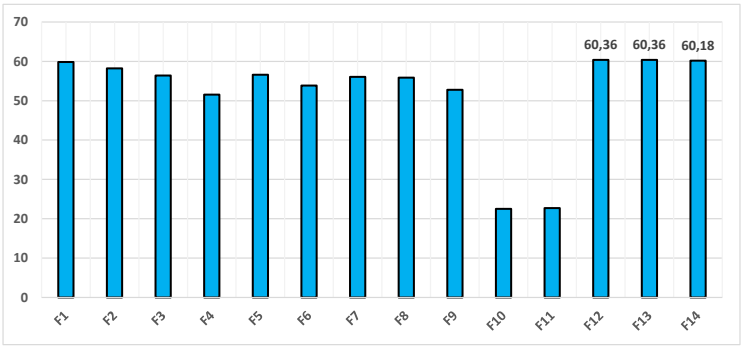

Figure 2. Average accuracies for all feature sets across all training corpora (using $\lambda = 0.6$).

As can be seen, the majority of all feature sets are more or less equally strong, excepting F_{10} and F_{11}, which seem to be useless for VEBAV (at least for $\mathcal{C}_{\textit{PAN-Train}}$). Another observation is that mixing several feature sets increases the classification quality only slightly.

5.4 Experiment III: Single feature sets vs. feature set combinations

In this experiment we were curious to know, if using combinations of feature sets by applying majority-voting can outperform classifications based only on single feature sets. As a setup for this experiment, we picked out the six most promising feature sets $\{F_1, F_2, F_5, F_{12}, F_{13}, F_{14}\}$ and used them to construct a power set $\mathbb{P}$, which includes $2^6 = 64$ feature set combinations. Next, we removed those subsets $F_{\textit{comb}i} \in (\mathbb{P} \setminus \emptyset)$ comprising of an even number of feature set combinations, to enable a fair (non-random based) majority-voting and also to speed the classification process up a little by avoiding unnecessary runs. This led to $2^5 = 32$ suitable combinations $F_{\textit{comb}1}, F_{\textit{comb}2}, \ldots, F_{\textit{comb}32}$, where we applied each $F_{\textit{comb}}$ as an input for VEBAV regarding $\mathcal{C}_{\textit{PAN-Train}}$. We stored all combinations and their corresponding classification results in a list (sorted in descending order) and extracted out the top five results, given in Table 2.

It can be observed from Table 2 that applying majority-voting on feature set combinations gives negligible results ($61.8\% - 60.36\% = 1.44\%$).

5.5 Experiment IV: Obtaining corpus dependent parameters

Due to the fact that the classification scores regarding the *Experiments I-III* were relatively low, we decided in this experiment to learn individual parameters from each

$\boldsymbol{F}_{comb}$	**Accuracy**
$\{F_1, F_2, F_5, F_{12}, F_{14}\}$	$61,8\%$
$\{F_3, F_{12}, F_{13}\}$	$61,62\%$
$\{F_1, F_2, F_5, F_{13}, F_{14}\}$	$61,62\%$
$\{F_2, F_5, F_{12}, F_{13}, F_{14}\}$	$61,26\%$
$\{F_1, F_3, F_{12}\}$	$61,26\%$

Table 2: Five top performing feature set combinations.

corpus $\mathcal{C} \in \mathcal{C}_{DE}, \mathcal{C}_{DR}, \mathcal{C}_{EE}, \mathcal{C}_{EN}, \mathcal{C}_{GR}, \mathcal{C}_{SP}$. Therefore, we first applied VEBAV on each $\mathcal{C}$ to obtain individual λ scores. Since there where six corpora, we constructed six tables, where the rows denote $F_1, F_2, ..., F_{14}$ and the coloumns the 14 λ values. Then, we picked those six tuples (F_i, λ_j), which led to the maximum accuracy score in each table. The selected tuples are given in Table 3.

$\boldsymbol{\mathcal{C}}$	$(\boldsymbol{\mathcal{F}_i}, \lambda_j)$	**Accuracy**
$\mathcal{C}_{DE}$	$(F_{12}, 0.6)$	$73,68\%$
$\mathcal{C}_{DR}$	$(F_{12}, 0.8)$	$60,76\%$
$\mathcal{C}_{EE}$	$(F_{12}, 1)$	$63,75\%$
$\mathcal{C}_{EN}$	$(F_3, 0.6)$	75%
$\mathcal{C}_{GR}$	$(F_3, 0.2)$	65%
$\mathcal{C}_{SP}$	$(F_7, 1)$	$71,25\%$

Table 3: The most promising tuple for each training corpus.

5.6 Results for the test set

In order to evaluate VEBAV on the test set $\mathcal{C}_{PAN\text{-}Test}$ we used all relevant information, learned from the prior experiments. For the first evaluation we set as an input for VEBAV the generalized parameters $\lambda = 0.6$ and $\mathcal{F}_{12}$, learned from *Experiments I-II*. The results are given in Table 4.

$\boldsymbol{\mathcal{C}}$	**Accuracy**	***AUC*·*c*@1**
$\mathcal{C}_{DE}$	80%	$0,40248$
$\mathcal{C}_{DR}$	45%	$0,2445525$
$\mathcal{C}_{EE}$	65%	$0,3147625$
$\mathcal{C}_{EN}$	45%	$0,2309625$
$\mathcal{C}_{GR}$	50%	$0,262025$
$\mathcal{C}_{SP}$	55%	$0,2602875$

Table 4: Results regarding $\mathcal{C}_{PAN\text{-}Test}$, using generalized parameters.

In the second and last evaluation we used the individual parameters, learned in *Experiment IV*. The results are given in Table 5. As can be seen, it makes sense to use individual parameters over generalized parameters, but one may pay the price of overfitting.

$\mathcal{C}$	$(\mathcal{F}_i, \lambda_j)$	**Accuracy**	***AUC·c*@1**
$\mathcal{C}_{DE}$	$(F_{12}, 0.6)$	80%	$0,40248$
$\mathcal{C}_{DR}$	$(F_{12}, 0.8)$	55%	$0,30569$
$\mathcal{C}_{EE}$	$(F_{12}, 1)$	55%	$0,25801125$
$\mathcal{C}_{EN}$	$(F_3, 0.6)$	80%	$0,4146$
$\mathcal{C}_{GR}$	$(F_3, 0.2)$	70%	$0,362425$
$\mathcal{C}_{SP}$	$(F_7, 1)$	55%	$0,28072275$

Table 5: Results regarding $\mathcal{C}_{PAN\text{-}Test}$, using individual parameters.

6 Conclusion & future work

In this paper we presented a simple, scalable and fast authorship verification scheme for the Author Identification task within the PAN-2014 competition. Our method provides a number of benefits as for instance language independence (at least for Indo-European languages) and also independence of linguistic resources (e.g. ontologies, thesauruses, language models, etc.). A further benefit is the low runtime of the method, since there is no need for deep linguistic processing like POS-tagging, chunking or parsing. Another benefit is that the involved components within the method can be replaced easily as for example the classification method, the acceptance-threshold or the feature categories including their parameters. Moreover, the components can be extended or combined e.g. through ensemble-techniques (combination of several style deviation methods).

Unfortunately, besides benefits our approach includes several pitfalls, too. One of the biggest challenges, for example, is the inscrutability of the methods parameter-space, due to the fact that the number of possible configuration settings is near infinite. Such settings include for instance the λ parameter for the involved distance function, the values for n and k (n-grams, k-prefixes/suffixes) but most of all the weight (ω) and tolerance (τ) parameters that influence to classification quality. Due to the complexity of our scheme we could only perform a small number of experiments to obtain at least an optimal λ and the most promising feature sets.

Another challenge that remains unsolved is how to optimize the probability score ρ determined in the decision determination step, as this value has also a strong influence on the resulting AUC*c@1 scores. Hence, further tests are essential for the applicability of the scheme.

7 Acknowledgements

This work was supported by the CASED Center for Advanced Security Research Darmstadt, Germany funded by the German state government of Hesse under the LOEWE programme (http://www.CASED.de).

References

1. Cha, S.H.: Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. International Journal of Mathematical Models and Methods in Applied Sciences 1(4), 300–307 (2007), http://www.gly.fsu.edu/ parker/geostats/Cha.pdf

2. Koppel, M., Schler, J.: Authorship Verification as a One-Class Classification Problem. In: Proceedings of the twenty-first international conference on Machine learning. pp. 62–. ICML '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1015330.1015448
3. Stamatatos, E.: A Survey of Modern Authorship Attribution Methods. J. Am. Soc. Inf. Sci. Technol. 60(3), 538–556 (Mar 2009), http://dx.doi.org/10.1002/asi.v60:3
4. Tax, D.M.J.: One-Class Classification. Concept Learning In the Absence of Counter-Examples. Ph.D. thesis, Delft University of Technology (2001), http://www-ict.ewi.tudelft.nl/~davidt/thesis.pdf