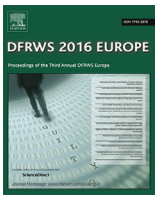
Digital Investigation 16 (2016) S33–S43

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



DFRWS 2016 Europe — Proceedings of the Third Annual DFRWS Europe

Authorship verification for different languages, genres and topics

Oren Halvani*, Christian Winter, Anika Pflug

Fraunhofer Institute for Secure Information Technology SIT, Rheinstr. 75, 64295 Darmstadt, Germany [1]

Keywords:
Digital text forensics
Intrinsic authorship verification
One-class-classification
Cross-genre
Cross-topic

A B S T R A C T

Authorship verification is a branch of forensic authorship analysis addressing the following task: Given a number of sample documents of an author $\mathscr{A}$ and a document allegedly written by $\mathscr{A}$, the task is to decide whether the author of the latter document is truly $\mathscr{A}$ or not. We present a scalable authorship verification method that copes with this problem across different languages, genres and topics. The central concept of our method is a model, which is trained with Dutch, English, Greek, Spanish and German text documents. The model sets for each language specific parameters and a threshold that accepts or rejects the alleged author as $\mathscr{A}$. The proposed method offers a wide range of benefits, e.g., a universal (static) threshold for each language and scalability regarding almost any involved component (classification function, ensemble strategy, features, etc.). Furthermore, the method benefits from low runtime due to the fact that no natural language processing techniques nor other computationally-intensive methods are involved. In our experiments, we applied the method on 28 test corpora including 4525 verification cases across 16 genres and a huge number of mixed topics, where we achieved competitive results (75% median accuracy). With these results we were able to outperform two state-of-the-art baselines, given the same training and test corpora.

Introduction

Forensic authorship analysis forms a branch of digital forensics with many application scenarios. There are a lot of real-world cases where a document has a certain alleged author, but the authorship is disputed by another party. Examples are university theses suspected that they have been written by a ghostwriter (Mothe et al., 2015), fictive insurance claims invented by a field agent of an insurance company, a forged last will or sock puppet detection (also known as multiple account detection) (Afroz et al., 2014).

The underlying forensic task is *authorship verification*. Authorship verification can also be applied for deciding whether two documents originate from the same author, e.g., for ascertaining whether two illegal online services, like illegal drug stores or illegal file distribution platforms, are operated by the same person. Due to the importance of authorship verification for forensic trials, this paper focuses on this task.

Another kind of forensic authorship analysis is *authorship profiling*, which determines author specific attributes such as gender, age or regional and social background of an unknown author, e.g., a blackmailer. A third variant is *authorship attribution* (also known as authorship identification), which tries to determine the true author in cases where several people are suspected as author of a document like a threatening letter, a terroristic proclamation, or a book published under a pseudonym.

* Corresponding author.
E-mail addresses: Oren.Halvani@SIT.Fraunhofer.de (O. Halvani), Christian.Winter@SIT.Fraunhofer.de (C. Winter), Anika.Pflug@SIT.Fraunhofer.de (A. Pflug).
[1] www.SIT.Fraunhofer.de.

http://dx.doi.org/10.1016/j.diin.2016.01.006

Authorship attribution (AA) deals with the problem of attributing a given anonymous text to one author, given a set of candidate authors for whom text samples of undisputed authorship are available (Stamatatos, 2009). The set of all authors with their corresponding text samples is usually referred to as *reference set*, which is analyzed regarding the writing style of the candidates in order to compare it to the writing style of the anonymous author. From this, a decision about the true author is made. If it is guaranteed that the reference set contains the true author, the AA task is considered to be a closed-set and otherwise an open-set problem. The majority of existing studies focuses on closed-set problems (Stolerman et al., 2014), which are known to be easier to solve than open-set problems (Potha and Stamatatos, 2014).

Authorship verification (AV) addresses the problem to determine whether a given text was written by a certain author $\mathscr{A}$ or not (Stamatatos, 2009). The reference set in AV comprises only text samples from the supposed author $\mathscr{A}$, that can be analyzed for distinct features that describe the writing style of $\mathscr{A}$. If it differs significantly from the writing style of the given text, the authorship is considered false, otherwise true. AA can be treated as a sequence of AV problems regarding the given candidate authors (Koppel et al., 2012) and should be solvable if the verification problem is solved (Koppel and Winter, 2014). Conversely, AV represents a specialized task of AA with an open set of candidate authors. In this regard, AV can be understood as a *one-class classification problem* (Stein et al., 2008). This means a text is either classified as part of the given class, i.e. attributed to the supposed author, or classified as an outlier or negative example of the class (Tax, 2001).

If modeled as a one-class classification task, AV is a more difficult problem. This is mostly due to the challenge of determining boundaries between class elements and outliers without negative examples or at least exhaustive and representative positive examples (Koppel and Schler, 2004). In spite of its difficulty, AV is a rewarding topic because it often occurs as a task in practice, and it can also aid in other tasks as, for example, intrinsic plagiarism detection (Stein et al., 2008).

Challenges

AV, as a special case of AA, faces partially the same challenges. Similar to AA in general, some obstacles are, according to Stamatatos (2009):

- Sparse, unequally distributed, non-representative or difficult text sources (e.g., colloquial phrasing, short texts, noise from careless or author-typical mistakes).
- Coping with influences from topic, genre, time period or language of the document.
- Accuracy of involved tools (e.g., natural language processing tools).
- Finding appropriate features for the given task.

AV as a one-class classification problem faces challenges that are common to classification tasks. However, the biggest obstacle in AV is determining a universal threshold for separating genuinely authored texts from negative examples. In comparison, general AA has the advantage of balancing this decision between available positive and negative examples. In theory, negative examples can also be found for AV because all texts which were definitely not written by the supposed author represent outliers. However, due to the large amount of these possible examples, selecting those which are most representative or encompassing is difficult (Koppel and Schler, 2004). Hence, in comparison to AA, it is also more challenging to determine the most important text features for distinguishing authors. The selected features might only work best with the gathered negative samples and fail in other cases.

Furthermore, the decision boundary or threshold has to be adjusted to the application area of the verification method. It might be desirable to have a very liberal or conservative decision. This means the method either identifies a lot of same-author texts while also falsely identifying negative examples, or it only categorizes the most certain texts as same-author.

Extrinsic versus intrinsic methods

According to Stamatatos et al., an AV method can be either *intrinsic* or *extrinsic* (Stamatatos et al., 2014). The authors explain that intrinsic methods rely only on the questioned document $\mathscr{D}_{\mathscr{A}_?}$ and the reference set $\mathbb{D}_{\mathscr{A}}$ of the known author $\mathscr{A}$ for deciding the verification task. In contrast, extrinsic methods make use of additional documents by other authors in order to transform AV from a one-class-classification task to a binary classification task.

Our proposed AV method is an intrinsic method, since we do not use additional texts of other authors for deciding whether, the $\mathscr{D}_{\mathscr{A}_?}$ has been written by $\mathscr{A}$ or not.

Contribution

We propose a new intrinsic AV method that offers a number of contributions. Our method provides a universal threshold per language, 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 flexible and can be extended incrementally in order to handle new languages, genres or features. Our method does not involve error-prone natural language processing techniques nor machine learning libraries (e.g., Weka, RapidMiner, Scikit or Shogun), which often encapsulate the classification task as a black box. Furthermore, the method is compact and fully transparent and thus, can be reimplemented easily by the community. Moreover, it features a low computational complexity (on average, few seconds per case) compared to other existing approaches (e.g., listed in Stamatatos et al. (2014)). The evaluation of the proposed method on 28 test corpora distributed over five languages, 16 genres, and mixed topics shows, that the performance is stable in terms of accuracy, even for uncommon AV scenarios such as diploma theses or cooking recipes.

Related work

In recent years extrinsic AV approaches have been quite successful. A prominent example is the *Impostors Method* (IM), proposed by Koppel and Winter (2014). Given the questioned document $\mathscr{D}_{\mathscr{A}_?}$ and a reference document $\mathscr{D}_{\mathscr{A}}$, the method collects *impostor* texts for both of them (e.g., by using a search engine) and thus transforms the one-class classification problem of AV into a multi-class classification problem. Afterward it measures text similarity over several iterations with different subsets of character *n*-gram features. If the input text pair often enough has the highest similarity value compared to pairings with impostors, $\mathscr{D}_{\mathscr{A}_?}$ is assumed to be written by $\mathscr{A}$. An important part of the approach is the *impostor* selection. According to Koppel and Winter (2014), the chosen texts need to have just the right degree of similarity to the original document in order to not skew the result. This becomes problematic when two questioned documents differ in genre or topic (Koppel and Winter, 2014). On the other hand, IM performs well in practice and offers easily calculable features, that can be generated reliably in different languages.

The winners of the PAN competitions in 2013 and 2014 both use slight modifications of IM within their approaches (Stamatatos et al., 2014). In 2013, Seidman (2014) modified the method to optimize its parameters (e.g., feature type, number of iterations and acceptance threshold) depending on language in a training phase. The implementation also allows for a comparison of $\mathscr{D}_{\mathscr{A}_?}$ to a collection of reference texts $\mathbb{D}_{\mathscr{A}}$. Overall, the approach performs very well on the PAN 2013 corpus (Seidman, 2014).

In 2014, Khonji and Iraqi (Khonji and Iraqi, 2014) introduced another modification for IM. Modifications of the original method include an enhancement of the employed similarity measure and the inclusion of a big variation of features (e.g., word, character and POS tag *n*-grams). The implementation of the method includes a training phase for parameter optimization. However, according to the developers, it could be improved with regard to genre and language characteristics (Khonji and Iraqi, 2014). The discussed method scores consistently high on the PAN 2014 corpus but exhibits a long runtime (Stamatatos et al., 2014).

In contrast to extrinsic approaches, intrinsic methods also perform very well. One example of a simple but successful intrinsic approach is the *profile based method* proposed by Potha and Stamatatos (2014), which itself is based on Keselj et al. well-known AA method (Keselj et al., 2003). The *profile based method* uses features of the most frequent character *n*-grams and a dissimilarity measure to make a decision on the AV task via an acceptance threshold. The approach once again includes a training phase for parameter optimization for different languages. The method performs similarly well on the PAN 2013 corpus (Potha and Stamatatos, 2014) compared to the previously mentioned IM variation by Seidman (2014).

The second best overall score at PAN 2014 is achieved by another intrinsic method proposed by Fréry et al. (2014) that uses a CART algorithm (classification and regression trees) to decide if $\mathscr{D}_{\mathscr{A}_?}$ is written by $\mathscr{A}$ or not. Here, different verification problems within one language are used as a training set. Their features provide a measure of dissimilarity between known and unknown texts regarding every used feature type (e.g., word, character frequency, vocabulary and punctuation features). Notable is the integration of different genres in the training corpora. This allows for models that have been learned on genre as well as language. Besides scoring high at PAN 2014, the method also exhibits an efficient runtime (Fréry et al., 2014).

Another intrinsic AV method that was ranked fourth at PAN 2014 is the approach of Moreau et al. which is originally described under the name *robust approach* in Moreau et al. (2014). The core idea of their method is to measure the similarity between the set of known documents and the unknown document for every case in the training set, and then to determine the optimal threshold θ based on all the training cases. In testing mode, the same similarity measure is computed, and the answer is "Yes" if and only if the score is higher than θ. Moreau et al. use a slightly improved version of this idea. Instead of a single similarity score, five values are used:

- Two *similarity* scores, using a directed variant of the Jaccard similarity applied to character 4-grams (computed from the perspective of both documents).
- Two *divergence* scores, measuring the mean of the difference of the relative frequencies (computed from the perspective of both documents).
- The number of known documents for this case.

These values are the features provided to a classification learning algorithm (decision trees or SVM, with several variants using different parameters), trained on all training cases for one language. Parameters such as character n-gram minimum frequency and learning algorithm variant are determined with a genetic algorithm on the training set.

These examples show that intrinsic AV methods have to be considered competitive. They might also be able to compensate for weaknesses found in extrinsic approaches like genre and topic dependence. In the context of the discussed research, this paper proposes an intrinsic approach, which is compared in the experiments against the *robust approach* (Moreau et al., 2014) of Moreau et al. as well as the *profile-based* AV method of Potha and Stamatatos, 2014.

We would like to point out that our approach builds on our previous AV scheme (Halvani and Steinebach, 2014) in terms of notation and a part of the used features. However, the algorithm in Halvani and Steinebach (2014) entirely differs from our current approach.

Features

Features enable a simple, but efficient and effective heuristic to approximate individual writing styles and thus, to distinguish authors from each other.

Many attempts have been made to define and structure the variety of features across different research fields such as linguistics, psychology, computer science or mathematics. In his comprehensive survey (Stamatatos, 2009), for example, Stamatatos distinguishes between character, lexical, syntactic, semantic and application-specific *feature types*.

Table 1
The nine feature categories F_1, F_2, … F_9 used by our method by applying each F_i on a given document $\mathscr{D}$.

Feature category	Feature description & example	Parameters
F_1: Punctuation *n*-grams	A sequence of *n* consecutive punctuation marks (commas, hyphens, etc.) taken from $\mathscr{D}$ after reduction to punctuation characters. `This.is/a:sample-text` $\xrightarrow{n=3}$ `(./:, /:-)`	$n \in \{1, 2, \ldots, 10\}$
F_2: Character *n*-grams	A sequence of *n* consecutive characters in $\mathscr{D}$. `This is a sample text` $\xrightarrow{n=3}$ `(Thi, his, is␣, s␣i, ␣is, is␣, s␣a,…)`	$n \in \{1, 2, \ldots, 10\}$
F_3: *n*% frequent tokens	The *n*% most frequently occurring tokens in $\mathscr{D}$.	$n \in \{5, 10, \ldots, 50\}$
F_4: Token *k*-prefixes	The first *k* characters of a token. `This is a sample text` $\xrightarrow{n=2}$ `(Th, is, sa, te)`	$k \in \{1, 2, 3, 4\}$
F_5: Token *k*-suffixes	The last *k* characters of a token. `This is a sample text` $\xrightarrow{n=2}$ `(is, is, le, xt)`	$k \in \{1, 2, 3, 4\}$
F_6: Token *k*-prefix *n*-grams	The first *k* characters of each token within a token *n*-gram. `This is a sample text` $\xrightarrow{n=2}$ `(This␣is, is␣a, a␣sample, sample␣text)` $\xrightarrow{k=2}$ `(Th␣is, sa␣te)`	$n \in \{2, 3, 4\}$, $k \in \{1, 2, 3, 4\}$
F_7: Token *k*-suffix *n*-grams	The last *k* characters of each token within a token *n*-gram. `This is a sample text` $\xrightarrow{n=2}$ `(This␣is, is␣a, a␣sample, sample␣text)` $\xrightarrow{k=2}$ `(is␣is, le␣xt)`	$n \in \{2, 3, 4\}$,$k \in \{1, 2, 3, 4\}$
F_8: *n*-prefixes–*k*-suffixes	The first *n* and last *k* characters of a token. `This is a sample text` $\xrightarrow{n,k=2}$ `(Th␣is, is, sa␣le, te␣xt)`	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. `This is a sample text` $\xrightarrow{n=3,k=2}$ `(his␣is, ple␣te)`	n,$k \in \{1,2,3,4\}$

Regarding our approach we consider the more fine-grained distinction of the so-called *feature categories*. A feature category F_i falls into at least one of the feature types, defined by Stamatatos. In Table 1 we list nine feature categories used by our method. A frequent word used in Table 1 is the term *token(s)*, which we define as follows: Tokens represent strings, separated by exactly one space character (no tabs, newlines or other blank characters), where each token consists only of letters (of the given language) or punctuation marks or both. Any other string in a given document $\mathscr{D}$ is not considered as a token.

An interesting observation regarding F_3 is that the $n \approx 50\%$ frequent tokens in $\mathscr{D}$ represent mostly function words such as *although*, *since* or *while*. Hence, there is no need for any word-lists (regardless which language is given) to explicitly capture predefined function words. Function words have been used widely in the field of AV (Luyckx and Daelemans, 2008; Dinu and Popescu, 2009; Stolerman, 2015) as well as AA (Gamon, 2004; Zhao et al., 2005; Argamon and Levitan, 2005), and are often mentioned to work effectively.

As can be seen in Table 1 each feature category F_i is parametrized by *n*, *k* or both, given the listed ranges. These parameters have an impact regarding in which feature type they fall. For example a character 3-gram covers by default the *character* feature type. However, there are character 3-grams such as *any*, *not* or *the*, which cover also the *lexical* feature type. This is the reason why we make the more fine-grained distinction of *feature categories*.

With regard to our method we only make use of tokenization and simple regular expressions in order to extract features from texts, so that other more sophisticated natural language processing (NLP) tools such as lemmatizers, part-of-speech taggers or parsers are not taken into account. This consideration leads to some advantages as for instance:

- NLP tools (for instance syntactic parsers) are often bound to only one language, while this restriction, in general, does not apply to tokenization or regular expressions.
- NLP tools require different linguistic resources such as model files, dictionaries or other external knowledge (e.g., WordNet), while for tokenization or regular expressions only the text itself is required.
- NLP tools are very often based on machine learning algorithms and/or complex statistical modeling methods.[2] As a consequence of the nested algorithms involved in many NLP tools, the time complexity is often very high. In contrast, tokenization or regular expressions are faster and are already built-in in modern programming languages.

Regarding the last point we would like to underline that the features extraction process for each F_i listed in Table 1 from a text, requires (in its most simple implementation) linear time complexity. However, when using sophisticated string matching methods, as for instance Aho and Corasick (1975), the runtime can be reduced significantly.

Corpora

In this section we give an overview of all training and test corpora used in our experiments. First, a brief explanation is given why finding publicly available AV corpora is challenging. Next, we describe the structure and uniform format that all of our involved corpora follow. Afterward, we introduce the training and test corpora and their corresponding statistics. Finally, we explain the preprocessing steps that were partially applied on the corpora.

[2] For example hidden Markov models or conditional random fields.

Corpora acquisition and challenges

One of the biggest challenges in the field of AV is to find standardized, suitable and publicly available corpora. Such corpora would make it much easier to reproduce and compare verification results.

A major reason why AV corpora are rarely found on the Internet might be due to copyright issues, since authors generally must agree if one wants to redistribute their texts. This, however, would imply tremendous effort in terms of organization, time and also money expenses. Until today only few attempts have been made to address this problem. The most prominent attempt in this context is PAN (Stamatatos et al., 2015), which is an evaluation lab on uncovering plagiarism, authorship, and social software misuse. The PAN organizers offer researchers the possibility to participate in a number of shared tasks (e.g., authorship attribution, authorship verification, author clustering or author diarization) and to compare results of submitted software runs against those of other participants. PAN does not only host these tasks but also provides publicly accessible corpora, which play a role in this paper regarding training and testing.

Corpus format

Each training or testing corpus used by our method follows a specific format, which has been adapted from the PAN AV corpora. A corpus $\mathscr{C}$ comprises of a set of problems $\{\rho_1, \rho_2, \ldots\}$. Each problem $\rho = (\mathscr{D}_{\mathscr{A}_?}, \mathbb{D}_{\mathscr{A}}, \alpha)$ consists of a questioned document $\mathscr{D}_{\mathscr{A}_?}$, a set of known documents $\mathbb{D}_{\mathscr{A}}$ and the answer $\alpha \in \{Y,N\}$ regarding the true authorship, where Y denotes the true and N the false authorship. Note that this notation is important for our proposed method.

The number of known documents in each ρ varies from one to ten documents. The lengths of the texts in all corpora vary between a few hundred and a few thousand words. Furthermore, the distribution of true and false authorships regarding all involved corpora is mostly equal and in very few cases near-equal. Each ρ contains texts from the same language and all problems in $\mathscr{C}$ are made of one language, which can be Dutch, English, Greek, Spanish or German. Each $\mathscr{C}$ has a unique naming convention starting with the prefix *Tr* or *Te* that denote whether $\mathscr{C}$ is a training or a test corpus. Next, two upper cased letters indicate its language (inspired by top-level domains abbreviation), followed by a dash and the identifier of the corpus, e.g., the second-level-domain, where the texts stem from. As a matter of course, all training and test corpora are entirely disjunct.

Training corpora

All training corpora used in our approach serve unexceptional to learn the model which consists of parameters and thresholds. The majority of them are the PAN AV training corpora from the years 2013 and 2014 (Stamatatos et al., 2014; Juola and Stamatatos, 2014). Each PAN AV corpus falls into one of the three categories *training corpus*, *test corpus* or *early bird corpus*. However, we excluded the last corpus category from our approach, since they are subsets of the test corpora. The languages covered by the PAN AV training corpora as well as their genres can be viewed in Table 2. A more detailed picture of the internals of the PAN AV is given in Stamatatos et al. (2014) and Juola and Stamatatos (2014).

Table 2
Training corpora, used in our experiments.

	$\mathbb{C}$	$\mathscr{C}$	$\lvert\mathscr{C}\rvert$	Genre	Source
Dutch	$\mathbb{C}_{nl}$	TrNL-PAN14es	96	essays	Stamatatos et al., 2014
		TrNL-PAN14re	100	reviews	Stamatatos et al., 2014
		TrNL-Trouw	200	news articles	self-compiled
English	$\mathbb{C}_{uk}$	TrUK-PAN14es	200	essays	Stamatatos et al., 2014
		TrUK-PAN14no	100	novels	Stamatatos et al., 2014
		TrUK-PAN13	10	textbooks	Juola and Stamatatos, 2014
		TrUK-Telegraph	200	news articles	self-compiled
Greek	$\mathbb{C}_{gr}$	TrGR-PAN14	100	articles	Stamatatos et al., 2014
		TrGR-PAN13	20	articles	Juola and Stamatatos, 2014
		TrGR-Tovima	70	articles	self-compiled
Spanish	$\mathbb{C}_{es}$	TrES-PAN14	100	articles	Stamatatos et al., 2014
		TrES-PAN13	6	editorials/ fiction	Juola and Stamatatos, 2014
		TrES-ElPais	200	news articles	self-compiled
German	$\mathbb{C}_{de}$	TrDE-Amazon	100	product reviews	self-compiled
		TrDE-D120	60	forum postings	self-compiled
		TrDE-Gutenberg	100	novels	self-compiled
		TrDE-Zeit	200	news articles	self-compiled

In order to improve the generalizability of our method, we decided to compile additional training and test corpora, besides the PAN AV training corpora. For this, we gathered texts from different sources such as news portals, forums, e-commerce platforms or free eBook distribution websites. The entire list of involved training corpora is given in Table 2.

Test corpora

The test corpora used in our experiments serve exclusively to evaluate the proposed AV method. In analogy to the PAN AV training corpora we used all the PAN AV test corpora from 2013 to 2014 (Stamatatos et al., 2014; Juola and Stamatatos, 2014). Moreover, we used other existing corpora such as the *Extended-Brennan-Greenstadt Adversarial Corpus* (Brennan et al., 2012), (denoted as TeUK-Drexel) or Koppel's *Blog Authorship Corpus* (Schler et al., 2006) (denoted as TeUK-KoppelBlog), which were modified to match the PAN AV corpus format. For the TeUK-Drexel corpus we used a small fraction (four documents) of the training data per author. For TeUK-KoppelBlog corpus we used only a part (2000 authors) of the entire corpus (19,320 authors), where we concatenated different texts for each author in order to form a mixed topic sample set of known documents. In addition to these we also compiled additional corpora,[3] which partially coined from genres that are

[3] The test corpora are available under http://bit.ly/1OjFRhJ.

uncommon in the field of AV (i.e. cooking recipes). Furthermore, we also compiled corpora from off-line data such as diploma theses written by our students, e-mails at our research institute as well as articles of purchased technical magazines.[4]

It should be highlighted that the majority of the involved test corpora include documents with mixed topics (e.g., TeDE-Amazon, TeDE-Mails or TeUK-Reddit). More precisely, within the problems in these corpora it often occurs that the known documents of the true author are coined from different topics. In addition to that, the topic of the unknown document sometimes also differs from the topics of the known documents. As a consequence, the AV task is even more challenging if the unknown document and the known documents were written by the same author. Note that due to the huge number of different topics it was not possible to add the info to Table 3.

Preprocessing

Regarding our self-compiled corpora, which have been mostly downloaded from the Internet, extensive preprocessing was necessary to make sure the data is suitable for the AV task. First, we removed duplicated and near-duplicated documents from the raw data. To achieve this we tokenized all documents into sets of tokens, in order to apply a set similarity function on each pair (X,Y). Here X refers to a token set of one document and Y to a token set of another document. We decided to use the overlap coefficient as a set similarity function, which (according to Liao et al. (1998)) is defined as follows:

$$\text{overlap}(X,Y) = \frac{|X \cap Y|}{\min(|X|,|Y|)} \quad (1)$$

Two documents were considered as near-duplicates, if their resulting overlap coefficient exceeded a specific threshold. We experimented with various thresholds (0.05, 0.1, 0.15, 0.25 and 0.5) and observed that two documents with a value below 0.25 are mostly similar in terms of function words, but not regarding their actual content. A similarity value above 0.25, however, indicates (based on our observations) that two documents share phrases, sentences or even passages with each other.

Next, we applied noise removal to all remaining documents. This involved removing mark-up tags, non-words, and digits as well as normalizing white space, e.g., by converting line breaks to spaces. Note that function words (also known as *stop words* or *non-content words*) and printable punctuation characters were excluded from this preprocessing, due to the fact that they form valuable features, as shown in the later experiments. After noise removal each document represents a long string, where each token is separated by exactly one space.

As a last step, we merged for each language the corresponding training corpora into a big training corpus $\mathbb{C}$. For example, the Dutch training corpus is described by $\mathbb{C}_{nl} = \text{TrNL} - \text{PAN14es} \cup \text{TrDU} - \text{PAN14re} \cup \text{TrNL} -$ Trouw. Our intention with this is to generate static feature category parameters as well as a unique threshold per language rather than per genre or topic.

[4] Note that due to copyright and privacy reasons, we are not able to provide these three test corpora.

Table 3
Test corpora, used in our experiments.

	$\mathscr{C}$	$\lvert\mathscr{C}\rvert$	Genre	Source
Dutch	TeNL-PAN14es	96	essays	Stamatatos et al., 2014
	TeNL-PAN14re	100	reviews	Stamatatos et al., 2014
	TeNL-RadarV	40	news articles	self-compiled
	TeNL-Trouw	200	news articles	self-compiled
English	TeUK-Drexel	44	research papers	Brennan et al., 2012
	TeUK-KoppelBlog	2000	blog posts	Schler et al., 2006
	TeUK-PAN13	30	textbooks	Juola and Stamatatos, 2014
	TeUK-PAN14es	200	essays	Stamatatos et al., 2014
	TeUK-PAN14no	200	novels	Stamatatos et al., 2014
	TeUK-Reddit	150	social news	self-compiled
	TeUK-Telegraph	100	news articles	self-compiled
Greek	TeGR-PAN13	30	articles	Juola and Stamatatos, 2014
	TeGR-PAN14	100	articles	Stamatatos et al., 2014
	TrGR-Sintagesparea	120	cooking recipes	self-compiled
	TeGR-Tovima	70	articles	self-compiled
Spanish	TeES-ElPais	200	news articles	self-compiled
	TeES-PAN13	25	editorials/ fiction	Juola and Stamatatos, 2014
	TeES-PAN14	100	articles	Stamatatos et al., 2014
	TeES-OcioZero	60	forum postings	self-compiled
	TeES-Rankia	100	forum postings	self-compiled
German	TeDE-Amazon	100	product reviews	self-compiled
	TeDE-CT	50	technical articles	self-compiled
	TeDE-D120	40	forum postings	self-compiled
	TeDE-Gutenberg	100	novels	self-compiled
	TeDE-Mails	24	e-mails	self-compiled
	TeDE-Recht	32	forum postings	self-compiled
	TeDE-Thesen	15	diploma theses	self-compiled
	TeDE-Zeit	200	news articles	self-compiled

Proposed verification method

In this section we present our AV method. First, we describe the training procedure that outputs the learned model, which forms the core of our method. Afterward, we describe the testing procedure that takes the learned model as an input in order to apply it on the test corpora.

Training procedure

The output of the training procedure is a model $\mathscr{M}$, which contains nine feature category configurations (the model created in our experiments from Section Experiments and evaluation is given in Table 4). Each feature category configuration denotes a table with five entries (for each language there is one entry). An entry for a specific language is a triple, consisting of the parameter n,

Table 4
Feature category configurations that form the model $\mathscr{M}$.

F_i	$\mathbb{C}_{nl}$	$\mathbb{C}_{uk}$	$\mathbb{C}_{gr}$	$\mathbb{C}_{es}$	$\mathbb{C}_{de}$
F_1	n = 2	n = 4	n = 3	n = 1	n = 1
	θ = 0.583	θ = 0.441	θ = 0.549	θ = 0.797	θ = 0.774
F_2	n = 10	n = 10	n = 10	n = 9	n = 10
	θ = 0.335	θ = 0.337	θ = 0.338	θ = 0.341	θ = 0.336
F_3	n = 45	n = 45	n = 50	n = 25	n = 45
	θ = 0.467	θ = 0.473	θ = 0.471	θ = 0.533	θ = 0.458
F_4	n = 4	n = 4	n = 2	n = 1	n = 3
	θ = 0.397	θ = 0.414	θ = 0.609	θ = 0.798	θ = 0.498
F_5	n = 3	n = 4	n = 4	n = 2	n = 1
	θ = 0.500	θ = 0.419	θ = 0.419	θ = 0.697	θ = 0.816
F_6	n = 3	n = 3	n = 4	n = 3	n = 4
	k = 2	k = 2	k = 2	k = 3	k = 2
	θ = 0.339	θ = 0.344	θ = 0.335	θ = 0.334	θ = 0.335
F_7	n = 3	n = 4	n = 4	n = 3	n = 4
	k = 2	k = 2	k = 3	k = 3	k = 2
	θ = 0.343	θ = 0.336	θ = 0.333	θ = 0.334	θ = 0.335
F_8	n = 3	n = 1	n = 2	n = 2	n = 4
	k = 4	k = 4	k = 4	k = 4	k = 4
	θ = 0.340	θ = 0.364	θ = 0.350	θ = 0.352	θ = 0.339
F_9	n = 4	n = 4	n = 4	n = 4	n = 4
	k = 3	k = 2	k = 4	k = 2	k = 3
	θ = 0.337	θ = 0.344	θ = 0.336	θ = 0.344	θ = 0.338

the parameter k (depending on the considered feature category F_i) and the associated threshold θ.

The following four steps describe in detail how the training is performed in order to construct a configuration for a specific F_i, given the training corpora. This procedure is performed for each corpus $\mathbb{C} \in \{\mathbb{C}_{nl}, \mathbb{C}_{uk}, \ldots, \mathbb{C}_{de}\}$ and each feature category $F_i \in \{F_1, F_2, \ldots, F_9\}$ in order to obtain $\mathscr{M}$.

Step 1: construct feature vectors and calculate pairwise similarity scores

The goal of this step is the assessment of similarity between the questioned document $\mathscr{D}_{\mathscr{A}?}$ and the samples of known documents in $\mathbb{D}_{\mathscr{A}}$. For this we follow the profile-based paradigm mentioned in Potha and Stamatatos (2014) and concatenate all texts in $\mathbb{D}_{\mathscr{A}}$ into $\mathscr{D}_{\mathscr{A}}$. Afterward, we construct for each problem ρ the numeric feature vectors $\mathscr{F}_{\mathscr{A}?}$ and $\mathscr{F}_{\mathscr{A}}$ corresponding to $\mathscr{D}_{\mathscr{A}?}$ and $\mathscr{D}_{\mathscr{A}}$ respectively. Each feature vector has the form of $(\varphi(f_1), \varphi(f_2), \varphi(f_3), \ldots)$, where $\varphi(f_j)$ denotes a relative frequency describing how often an extracted feature f_j occurs among all other extracted features in $\mathscr{D}_{\mathscr{A}?}$ or $\mathscr{D}_{\mathscr{A}}$, given a feature category F_i. An example for $\varphi(f_j)$ regarding a document $\mathscr{D}$ and the token f_j = while is given as:

$$\varphi(f_j) = \frac{\text{Count of occurrences of while in } \mathscr{D}}{\text{Total count of occurrences of } F_i - \text{features in } \mathscr{D}} \tag{2}$$

After the vectors have been constructed, we use, inspired by Burrow's research work (Burrows, 2002), the *Manhattan distance* function:

$$\text{dist}(X, Y) = \sum_{j=1}^{n} |x_j - y_j|, \text{ with } X = \mathscr{F}_{\mathscr{A}?}, Y = \mathscr{F}_{\mathscr{A}} \tag{3}$$

The resulting distance value is converted (for normalization purposes) into a similarity score by applying the commonly used transformation:

$$\text{sim}(X, Y) = \frac{1}{1 + \text{dist}(X, Y)} \tag{4}$$

Any other order-reversing conversion from the distance to a similarity score would also be suitable. We denote a similarity score associated to a problem ρ by s_ρ.

Step 2: determining the acceptance threshold

The goal of this step is to determine the acceptance threshold for the authorship verification based on the training corpus $\mathbb{C}$. This threshold is calculated separately for each feature category and combination of feature category parameters. The threshold θ will be used to classify a problem with Y if $s_\rho > \theta$ or with N if $s_\rho \leq \theta$ holds (see the classification function (6) in Step 3).

We choose θ such that we get an *equal error rate* (EER) for the problems in $\mathbb{C}$. This means that the false positive rate (i.e. fraction of negative training problems erroneously labeled with Y) equals the false negative rate (i.e. fraction of positive training problems erroneously labeled with N) when classifying the training problems based on θ. Fig. 1 illustrates this idea. In this figure area A is the false negative rate and area A' is the false positive rate. Note that the threshold θ is not necessarily located at the intersection of the two probability density functions, but it is mostly very close to this intersection.

We have chosen the EER as criterion for θ since our performance measure (see Step 3) assigns equal weights to false positives and false negatives. Such a measure is typical for academic evaluations. In real world scenarios false positives or false negatives might be more severe, and then a different choice for θ might provide a more suitable trade-off between false positives and false negatives.

For balanced corpora as in our setting, the threshold θ for the EER is determined as median of all similarity scores for positive and negative examples. In pathological cases where multiple similarity scores are equal to the median, an exact EER is not possible, but θ determined as median (or as next smaller similarity score) provides an optimal approximation of the EER.

Step 3: determining the best Feature category parameters

We repeat Step 1 and Step 2 for each feature category F_i and all of its possible combinations for the parameters n and k (described in Table 1). In order to judge how these combinations perform, a performance measure is required. We decided to use the *accuracy* measure, formulated as:

$$\text{accuracy} = \frac{\text{number of correct answers}}{\text{number of all problems in } \mathbb{C}} \tag{5}$$

Here, *correct answers* refers to such problems, where the classification result matches the true answer. In order to obtain a binary result $\alpha \in \{Y,N\}$ we classify each $\rho \in \mathbb{C}$ based on its similarity score s_ρ from Step 1 and the determined threshold θ from Step 2. The classification function is defined as:

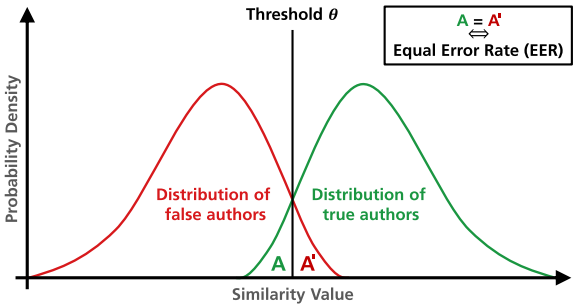


Fig. 1. Determining the acceptance threshold θ based on the EER.

$$\text{classify}(\rho) = \begin{cases} Y & \text{if } s_\rho > \theta \\ N & \text{otherwise} \end{cases} \quad (6)$$

As a result we get a table associating accuracies to all considered parameter combinations for all feature categories. Given this table, we select for each F_i that parameter tuple that leads to the maximal accuracy. The selected parameter tuples for the feature categories are stored as model $\mathscr{M}$. The resulting model of our experiments from Section Experiments and evaluation is given in Table 4, and the corresponding accuracies are provided in Table 5.

Step 4: Feature category ensembles

Once the model $\mathscr{M}$ is constructed, the next step is to construct an ensemble model $\mathscr{M}_{\mathscr{E}}$, which employs ensembles of feature categories. The benefit of ensembles is that they can outperform single feature categories, as shown in our experiments in Section Experiments and evaluation. An ensemble, denoted by $\mathscr{E}$ contains ℓ different feature categories with $\ell \in \{1, 3, 5, 7, 9\}$. Here, ℓ is uneven in order to apply a majority vote among all $F_i \in \mathscr{E}$.

In order to find the most promising ensemble for each language, represented by $\mathbb{C}$, we construct a set of all 256 possible ensembles (with an uneven number of feature categories) regarding the nine feature categories mentioned in Table 1. For each $\mathscr{E}$ we look up the feature category configurations in $\mathscr{M}$ for all involved $F_i \in \mathscr{E}$. Based on these configurations each problem $\rho \in \mathbb{C}$ is classified. The classification is performed for all $F_i \in \mathscr{E}$, where we collect all ℓ classification results (Y/N) regarding the authorship. Finally, we apply majority voting among all classification results associated to ρ in order to obtain the overall decision regarding the true/false authorship.

We calculate accuracies for all possible ensembles and pick the ensemble $\mathscr{E}_{\max}$, which leads to the maximum accuracy, given a training corpus $\mathbb{C}$. This procedure is performed for each of the five training corpora, such that each $\mathbb{C}$ is associated with exactly one $\mathscr{E}$. The result is a feature category ensemble model $\mathscr{M}_{\mathscr{E}}$, given in Table 6.

Testing procedure

In contrast to the training procedure, this procedure is considerably more compact. Given a feature category ensemble model $\mathscr{M}_{\mathscr{E}}$ and a test corpus $\mathscr{C}$, the only action that is performed is the classification of each problem in $\mathscr{C}$, based on the learned parameters and thresholds. The output of the testing procedure is a list of m answers regarding the m problems in $\mathscr{C}$. Based on this list and the true answers, the accuracy score is calculated for the evaluations.

Table 5
Experiment 1: accuracies in percent for all feature categories (training corpora results). The best accuracy per language is in bold. Additionally, the best median (in the median column and row respectively) is in bold.

F_i	$\mathbb{C}_{nl}$	$\mathbb{C}_{uk}$	$\mathbb{C}_{gr}$	$\mathbb{C}_{es}$	$\mathbb{C}_{de}$	Median
F_1	62.12	65.1	57.37	72.55	68.7	65.1
F_2	**71.21**	**73.33**	60	73.2	**74.78**	**73.2**
F_3	64.65	69.8	56.84	**77.78**	69.13	69.13
F_4	65.66	68.63	53.68	71.24	70.87	68.63
F_5	63.64	67.84	50.53	70.59	66.96	66.96
F_6	65.15	68.63	**67.37**	75.82	71.74	68.63
F_7	63.13	68.63	57.89	68.63	65.65	65.65
F_8	69.19	72.16	64.74	**77.78**	71.3	71.3
F_9	68.18	71.76	61.05	74.51	73.04	71.76
Median	65.15	68.63	57.89	**73.2**	70.87	

Experiments and evaluation

In this section we describe the experimental settings for training and testing and report our results and observations. The results are given in terms of the accuracy score, defined in Formula (5).

Experiment I – single feature categories

The goal of our first experiment is to figure out how all feature categories perform in comparison to each other. To achieve this, we apply the training procedure described in Section Proposed verification method on all training corpora and construct the model $\mathscr{M}$. The feature category parameters that form $\mathscr{M}$ are given in Table 4, while the accuracies we obtained with these parameters are listed in Table 5.

We observe from the "Median" column in Table 5 that character n-grams (F_2) perform slightly better, across all five languages, in comparison to the majority of the other feature categories. This observation strengthens the findings of several previous studies [e.g., Koppel and Winter, 2014; Potha and Stamatatos, 2014; Van Dam and Hauff, 2014], where character n-grams also have shown a high potential in comparison to other feature types. One possible reason for the effectiveness of character n-grams might be that they can (depending on the setting of n) cover mixtures of other feature categories such as function words or token prefixes/suffixes that also perform very

Table 6
Experiment 2: best performing ensembles per language (training corpora results).

Language	Accuracy (%)	Ensemble
Dutch	72.47	$\{F_2,F_8,F_9\}$
English	76.67	$\{F_1,F_2,F_3,F_7,F_8\}$
Greek	67.37	$\{F_6\}$
Spanish	83.33	$\{F_1,F_3,F_4,F_6,F_8\}$
German	78.04	$\{F_1,F_2,F_3,F_5,F_8\}$

well. Furthermore, Table 5 reveals that punctuation *n*-grams (F_1) perform worst. This can be explained by the fact that punctuation *n*-grams offer the smallest stylistic variance in comparison to the other eight feature categories. However, punctuation *n*-grams combined with other feature categories can be helpful, as can be seen in Experiment II.

We also observe from Table 5 that Greek seems to be the most challenging language for our method. However, it is not clear whether this is due to the small number of problems contained in the training corpus or due to the used feature categories. This question will be addressed in future work by compiling additional training corpora.

Experiment II – ensembles

The goal of this experiment is to figure out how well ensembles (as described in Step 4 of the training procedure) can outperform single feature categories, given the learned model $\mathcal{M}$. The results for the best performing ensembles of this experiment are summarized in Table 6.

For Greek the best ensemble consists of the single feature category F_6, i.e., no real ensemble performs better than this single feature category. However, for all other languages ensembles are superior to single feature categories. The largest improvement is 5.55Hence ensembles can lead to a significant increase of accuracy.

Experiment 3 – evaluation on test corpora

In this section we present the evaluation results of our AV method, as well as those of the following two baselines, given the 28 test corpora listed in Table 3.

Baselines

As a first baseline we use the *robust approach* AV method of Moreau et al. (2014) and as a second baseline the *profile-based* AV method of Potha and Stamatatos (2014), which were both described briefly in Section Related work. We asked the authors to evaluate their AV systems, given our training and testing corpora and to provide us their results. This strategy was chosen to make sure that our results are tested against the original implementations from the publications mentioned above. This is to prevent any inconsistencies between re-implementations and the original algorithms. Note that we asked also two other authors to participate in our study by evaluating our corpora on their systems, however, only the authors mentioned above agreed.

Evaluation results

The results for the baseline comparison are given in Table 7 and Table 8. There is no method which is the best for all corpora and all languages. However, our method achieves the highest accuracy in more cases than the baselines. Moreover, our median accuracy of 75% is well above the baselines.

A very interesting observation is that our method performs very well on news articles across all languages (except for Greek). This holds for the test corpora TeNL-RadarV (77.5% accuracy), TeNL-Trouw (75%), TeUK-Telegraph (77%), TeES-ElPais (84.5%) and TeDE-Zeit (76%). Note that these corpora contain a huge number of articles about mixed topics from the same authors, which indicates the model generalizes well regarding the topic.

One remarkable observation for Greek is that we achieved the best results on the cooking recipes corpus TrGR-Sintagesparea although this genre was not included in the training corpus. Furthermore, it can be concluded from Table 6 that Greek may show the worst overall performance. In contrast to other languages, none of the Greek test corpora was able to achieve strong results ($\geq$75%). However, the performance is stable among all tested genres although this might not be expected for an ensemble consisting of a single feature category. We observed that other ensembles (with more than one feature categories) do not perform better. The explanation of this phenomenon is subject for future work, as it is not yet clear whether this

Table 7
Evaluation results of our AV method compared to two baselines, given the 28 test corpora. The best result per corpus is in bold.

	Corpus	Our approach	Baselines	
			Moreau	Stamatatos
Dutch	TeNL-PAN14es	75	68.75	**89.58**
	TeNL-PAN14re	53	50	**57**
	TeNL-RadarV	77.5	**82.50**	67.50
	TeNL-Trouw	75	70	**76.50**
English	TeUK-Drexel	77.27	63.64	**81.82**
	TeUK-KoppelBlog	**73**	69.65	69.85
	TeUK-PAN13	**73.33**	66.67	63.33
	TeUK-PAN14es	**58**	48	55
	TeUK-PAN14no	**58.5**	50	54.50
	TeUK-Reddit	**75.33**	70	66.67
	TeUK-Telegraph	77	**84**	78
Greek	TeGR-PAN13	63.33	**73.33**	70
	TeGR-PAN14	**65.83**	48	65
	TrGR-Sintagesparea	**68.57**	55	55
	TeGR-Tovima	65	58.57	**67.14**
Spanish	TeES-ElPais	84.5	**85**	82
	TeES-Ocio	60	**71.67**	70
	TeES-PAN13	**76**	**76**	60
	TeES-PAN14	**72**	52	62
	TeES-Rankia	70	61	**81**
German	TeDE-Amazon	67	57	**70**
	TeDE-CT	76	74	**78**
	TeDE-D120	**82.5**	80	60
	TeDE-Gutenberg	**80**	73	71
	TeDE-Mails	83.33	**87.50**	75
	TeDE-Recht	**81.25**	**81.25**	68.75
	TeDE-Thesen	78.57	**92.86**	57.14
	TeDE-Zeit	76	73.50	**77**
Median Accuracy		**75**	70	69.3

Table 8
Evaluation results given as median accuracies per language. The best result per language is in bold.

Language	Our approach	Baselines	
		Moreau	Stamatatos
Dutch	**75**	69.38	72
English	**73.33**	66.67	66.67
Greek	65.42	56.79	**66.07**
Spanish	**72**	71.67	70
German	**79.29**	77	70.50

depends on the features themselves or on the texts (or even both).

Another observation is that German leads to the best overall result (median accuracy of 79.29% for our method) among all investigated languages. This is surprising, since the German training corpora cover only a subset of the genres existing in the test corpora, and the additional genres also include different registers, e.g., formal language (technical articles) and everyday language (e-mails). One possible reason for the good performance could be the morphological richness of the German language, which includes for instance long compound words. Hence a German text can provide more features in the same number of words as an English text, for example. Another reason (again, due the morphological nature) might be that German is a highly inflected language such that features like token suffixes are more diverse and therefore can discriminate between writing styles. This observation emphasizes again that the model generalizes well across different genres but also topics.

Another observation is that all tested methods perform bad for the three PAN-corpora TeNL-PAN14re, TeUK-PAN14es and TeUK-PAN14no. A deeper inspection of the corpora statistics (Stamatatos et al., 2014) reveals that the average number of known documents in each problem of the Dutch corpus TeNL-PAN14re is exactly one, and these documents have an average word count of 116.3 (this is much less than the abstract of this paper). As a result there are only few features available that have a strong impact regarding accepting or rejecting the given authorship. This explains the bad performance across all three methods for this test corpus. Regarding TeUK-PAN14es and TeUK-PAN14no we face a similar number of documents per problem, but a bigger average word count (833.2 for the former and 6104 for the latter). The PAN-organizers explain that the TeUK-PAN14no corpus is not focusing on a very small subgenre of speculative and horror fiction that includes extremely florid prose and an unusual vocabulary (Stamatatos et al., 2014). We infer from this that there are not many common features between $\mathscr{D}_{\mathscr{A}}$ and $\mathscr{D}_{\mathscr{A}_2}$, which again leads to bad results across the three methods.

Runtime

One important issue that is not considered in the tables or in the experiments is that our AV method performs quite modest regarding the runtime. Once the model $\mathscr{M}$ is constructed, the only tasks for testing that need to be performed are the computation of feature vectors and the similarities regarding the texts (both in linear runtime). For each of the involved test corpora we measure a runtime of $\approx$30 s, on an off-the-shelf laptop (Intel® Core™ i5-3210M processor, 16 GB RAM).

Conclusion and future work

We presented a simple, efficient and effective AV method that automatically verifies the alleged authorship of a text. The proposed method was evaluated on 28 test corpora (consisting of 4525 verification cases), distributed over five languages, 16 genres, and a huge number of mixed topics. It was shown that the method generalizes well (75% median accuracy), even when applied on genres that are uncommon in AV, as for instance cooking recipes, social news or diploma theses.

Our method generates a compact and transparent model that can be extended incrementally in order to work with new languages or other feature categories. The model consists of configuration parameters (settings and thresholds for each language and feature category), which are optimized automatically and thus, do not require a manual definition by the user. Another advantage is that the method does not require natural language processing techniques nor does it rely on any third party machine learning libraries. Instead, the method only makes use of feature extraction techniques (based on tokenization and regular expressions) and a distance measure, which ensures an easy reimplementation of the approach. In addition, the method has low computational complexity compared to other state-of-the-art approaches. In total, 4525 verification problems were verified in 274 s, which means on average, 0.06 s per problem.

Besides these benefits, the proposed method also leaves a lot of room for improvement. Currently, the strategy we use to optimize parameters in the training procedure is based on grid search. However, this strategy is known to perform poorly in practice (Bergstra and Bengio, 2012) and should, therefore, be replaced with more practical techniques (i.e. random search) in order to speed up the training.

Furthermore, we also plan to investigate the idea to find a single configuration instead of one configuration for each language. Initial results, based on three corpora, have shown that it is possible to successfully apply a set of parameters derived from one language to other, unseen languages (French, Polish and Swedish). We plan to further investigate this phenomenon and seek to explain our observations in future research.

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 (13N13065) and EWV (13N13500). At this point, we would also like to thank Erwan Moreau and Efstathios Stamatatos for providing us their results that helped us to compare our method.

References

Afroz S, Islam A, Stolerman A, Greenstadt R, Mccoy D. Doppelgaenger finder: taking stylometry to the underground. In: Security and Privacy (SP), 2014 IEEE Symposium on; 2014. p. 212–26. http://dx.doi.org/10.1109/SP.2014.21.

Aho AV, Corasick MJ. Efficient string matching: an aid to bibliographic search. Commun ACM 1975;18(6):333–40. http://dx.doi.org/10.1145/360825.360855. URL, http://doi.acm.org/10.1145/360825.360855.

Argamon S, Levitan S. Measuring the usefulness of function words for authorship attribution. In: ACH/ALLC 2005 – Conference Abstracts. 2nd ed. University of Victoria; 2005. p. 4–7 URL http://tomcat-stable.hcmc.uvic.ca:8080/ach/site/pdf.xq?id=162.

Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13:281–305. URL, http://dl.acm.org/citation.cfm?id=2188385.2188395.

Brennan M, Afroz S, Greenstadt R. Adversarial stylometry: circumventing authorship recognition to preserve privacy and anonymity. ACM Trans Inf Syst Secur 2012;15(3):12:1–12:22. http://dx.doi.org/10.1145/2382448.2382450. URL, http://doi.acm.org/10.1145/2382448.2382450.

Burrows J., 'Delta': a measure of stylistic difference and a guide to likely authorship, Lit Linguistic Comput 17(3) (2002) 267–287. arXiv:http://llc.oxfordjournals.org/content/17/3/267.full.pdf+html, doi:10.1093/llc/17.3.267. URL http://llc.oxfordjournals.org/content/17/3/267.abstract.

Dinu LP, Popescu M. Ordinal measures in authorship identification. Proc SEPLN 2009:62–6.

Fréry J, Largeron C, Juganaru-Mathieu M. UJM at CLEF in author identification – notebook for PAN at CLEF 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1180; 2014. p. 1042–8. URL, http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-FreryEt2014.pdf.

Gamon M. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In: Proceedings of Coling 2004, International Conference on Computational Linguistics; 2004. p. 611–7. URL, http://aclweb.org/anthology/C/C04/C04-1088.pdf.

Halvani O, Steinebach M. VEBAV – a simple, scalable and fast authorship verification scheme – notebook for PAN at CLEF 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1180; 2014. p. 1049–62. URL, http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-HalvaniEt2014.pdf.

Juola P, Stamatatos E. Overview of the author identification task at PAN 2013. In: Forner P, Navigli R, Tufis D, Ferro N, editors. Working Notes for CLEF 2013 Conference, Valencia, Spain, September 23–26, 2013. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1179; 2014. URL, http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-JuolaEt2013.pdf.

Keselj V, Peng F, Cercone N, Thomas C. N-gram-based author profiles for authorship attribution. In: Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING'03. Halifax, Nova Scotia, Canada: Dalhousie University; 2003. p. 255–64.

Khonji M, Iraqi Y. A slightly-modified GI-based author-verifier with lots of features (ASGALF) – notebook for PAN at CLEF 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1180; 2014. p. 977–83. URL, http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-KonijEt2014.pdf.

Koppel M, Schler J. Authorship verification as a one-class classification problem. In: Proceedings of the twenty-first international conference on Machine learning, ICML '04, ACM, New York, NY, USA; 2004. p. 489–95. http://dx.doi.org/10.1145/1015330.1015448.

Koppel M, Winter Y. Determining if two documents are by the same author. JASIST 2014;65(1):178–87. http://dx.doi.org/10.1002/asi.22954.

Koppel M, Schler J, Argamon S, Winter Y. The "fundamental problem" of authorship attribution. Engl Stud 2012;93(3):284–91. http://dx.doi.org/10.1080/0013838X.2012.668794.

Liao TW, Zhang Z, Mount CR. Similarity measures for retrieval in case-based reasoning systems. Appl Artif Intell 1998;12(4):267–88. http://dx.doi.org/10.1080/088395198117730.

Luyckx K, Daelemans W. Authorship attribution and verification with many authors and limited data. In: Proceedings of the 22nd International Conference on Computational Linguistics. COLING '08, vol. 1. Stroudsburg, PA, USA: Association for Computational Linguistics; 2008. p. 513–20. URL, http://dl.acm.org/citation.cfm?id=1599081.1599146.

Moreau E, Jayapal A, Vogel C. Author verification: exploring a large set of parameters using a genetic algorithm – notebook for pan at CLEF 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1180; 2014. p. 1092–103. URL, http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-MoreauEt2014.pdf.

Mothe J, Savoy J, Kamps J, Pinel-Sauvagnat K, Jones G, San Juan E, et al. Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association, CLEF'15, Toulouse, France, September 8–11, 2015. In: Proceedings, Lecture Notes in Computer Science. Springer International Publishing; 2015. URL, https://books.google.de/books?id=0l14CgAAQBAJ.

Potha N, Stamatatos E. A profile-based method for authorship verification. In: 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15–17, 2014, Proceedings. Springer International Publishing; 2014. p. 313–26. http://dx.doi.org/10.1007/978-3-319-07064-3_25.

Schler J, Koppel M, Argamon S, Pennebaker JW. Effects of age and gender on blogging. In: Computational Approaches to Analyzing Weblogs, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-03, Stanford, California, USA, March 27–29, 2006. Association for the Advancement of Artificial Intelligence; 2006. p. 199–205. URL, http://www.aaai.org/Library/Symposia/Spring/2006/ss06-03-039.php.

Seidman S. Authorship verification using the impostors method – notebook for PAN at CLEF. In: Forner P, Navigli R, Tufis D, Ferro N, editors. Working Notes for CLEF 2013 Conference, Valencia, Spain, September 23–26, 2013. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1179; 2014. URL, http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-Seidman2013.pdf.

Stamatatos E. A survey of modern authorship attribution methods. J Am Soc Inf Sci Technol 2009;60(3):538–56. http://dx.doi.org/10.1002/asi.v60:3.

Stamatatos E, Daelemans W, Verhoeven B, Potthast M, Stein B, Juola P, et al. Overview of the author identification task at PAN 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, CEUR-WS.org, vol. 1180; 2014. p. 877–97. URL, http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf.

Stamatatos E, Potthast M, Rangel F, Rosso P, Stein B. Overview of the PAN/CLEF 2015 evaluation lab. In: Mothe J, Savoy J, Kamps J, Pinel-Sauvagnat K, Jones GJ, San Juan E, et al., editors. Experimental IR Meets Multilinguality, Multimodality, and Interaction – 6th International Conference of the CLEF Association, CLEF'15, Toulouse, France, September 8–11, 2015, Proceedings. LNCS, vol. 9283. Springer; 2015. p. 518–38. http://dx.doi.org/10.1007/978-3-319-24027-5_49.

Stein B, Lipka N, Meyer zu Eißen S. Meta analysis within authorship verification. In: 19th International Workshop on Database and Expert Systems Applications (DEXA 2008), 1–5 September 2008, Turin, Italy. IEEE Computer Society; 2008. p. 34–9. http://dx.doi.org/10.1109/DEXA.2008.20. URL, http://doi.ieeecomputersociety.org/10.1109/DEXA.2008.20.

Stolerman A. Authorship verification. Ph.D. thesis. Drexel University; 2015. URL, http://search.proquest.com/docview/1679459529.

Stolerman A, Overdorf R, Afroz S, Greenstadt R. Breaking the closed-world assumption in stylometric authorship attribution. Revised Selected Papers. In: Peterson GL, Shenoi S, editors. Advances in Digital Forensics X – 10th IFIP WG 11.9 International Conference, Vienna, Austria, January 8–10, 2014. IFIP Advances in Information and Communication Technology, vol. 433. Springer; 2014. p. 185–205. http://dx.doi.org/10.1007/978-3-662-44952-3_13.

Tax DMJ. One-class classification: concept learning in the absence of counter-examples. Ph.D. thesis. Delft University of Technology; 2001. URL, http://www-ict.ewi.tudelft.nl/~{}davidt/thesis.pdf.

Van Dam M, Hauff C. Large-scale author verification: temporal and topical influences. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '14. New York, NY, USA: ACM; 2014. p. 1039–42. http://dx.doi.org/10.1145/2600428.2609504. URL, http://doi.acm.org/10.1145/2600428.2609504.

Zhao Y, Zobel J. Effective and scalable authorship attribution using function words. In: Lee G, Yamada A, Meng H, Myaeng S, editors. Information Retrieval Technology. Lecture Notes in Computer Science, vol. 3689. Springer Berlin Heidelberg; 2005. p. 174–89. http://dx.doi.org/10.1007/11562382_14.