Same or Different? Diff-Vectors for Authorship Analysis

SILVIA CORBARA, Scuola Normale Superiore, Pisa, Italy
ALEJANDRO MOREO and FABRIZIO SEBASTIANI, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy

In this article, we investigate the effects on authorship identification tasks (including authorship verification, closed-set authorship attribution, and closed-set and open-set same-author verification) of a fundamental shift in how to conceive the vectorial representations of documents that are given as input to a supervised learner. In "classic" authorship analysis, a feature vector represents a document, the value of a feature represents (an increasing function of) the relative frequency of the feature in the document, and the class label represents the author of the document. We instead investigate the situation in which a feature vector represents an unordered *pair* of documents, the value of a feature represents the absolute difference in the relative frequencies (or increasing functions thereof) of the feature in the two documents, and the class label indicates whether the two documents are from the same author or not. This latter (learner-independent) type of representation has been occasionally used before, but has never been studied systematically. We argue that it is advantageous, and that, in some cases (e.g., authorship verification), it provides a much larger quantity of information to the training process than the standard representation. The experiments that we carry out on several publicly available datasets (among which one that we here make available for the first time) show that feature vectors representing pairs of documents (that we here call *Diff-Vectors*) bring about systematic improvements in the effectiveness of authorship identification tasks, and especially so when training data are scarce (as it is often the case in real-life authorship identification scenarios). Our experiments tackle same-author verification, authorship verification, and closed-set authorship attribution; while DVs are naturally geared for solving the 1st, we also provide two novel methods for solving the 2nd and 3rd that use a solver for the 1st as a building block. The code to reproduce our experiments is open-source and available online.[1]

CCS Concepts: • **Computing methodologies → Supervised learning by classification**; **Natural language processing;**

Additional Key Words and Phrases: Supervised learning, vector-based representations, authorship analysis

[1] https://github.com/AlexMoreo/diff-vectors

The authors' work has been supported by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData.it, FAIR and ITSERR projects funded by the Italian Ministry of University and Research under the NextGenerationEU program. These authors' opinions do not necessarily reflect those of the funding agencies.

Authors' addresses: S. Corbara, Scuola Normale Superiore, Pisa 56126, Italy; email: silvia.corbara@sns.it; A. Moreo and F. Sebastiani, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa 56124, Italy; emails: {alejandro.moreo, fabrizio.sebastiani}@isti.cnr.it.

1556-4681/2023/09-ART12 $15.00
https://doi.org/10.1145/3609226

ACM Reference format:
Silvia Corbara, Alejandro Moreo, and Fabrizio Sebastiani. 2023. Same or Different? Diff-Vectors for Authorship Analysis. *ACM Trans. Knowl. Discov. Data.* 18, 1, Article 12 (September 2023), 36 pages.
https://doi.org/10.1145/3609226

1 INTRODUCTION

Recent years have seen an increased interest in automated *authorship analysis*, a set of tasks aiming to infer the characteristics of the author of a text of unknown or disputed paternity. Authorship analysis is concerned with inferring characteristics such as the gender [27], the age group [16], or the native language [45] of the author, among others; these subtasks usually go under the name of *author profiling* [3]. Alternatively, authorship analysis may be concerned with inferring the *identity* of the author; tasks in which this is the goal are collectively referred to as *authorship identification* tasks, and include ***authorship verification*** (**AV**) (AV—the task of predicting whether a given author is or not the author of a given anonymous text [44]), ***authorship attribution*** (**AA**) (AA—the task of predicting who among a given set of candidates is the most likely author of a given anonymous text [20, 28, 43]), and ***same-author verification*** (**SAV**) (SAV—the task of predicting whether two given documents are by the same, possibly unknown, author or not [29]). Authorship analysis has several applications, e.g., in supporting the work of philologists who try to identify the authors of texts of literary or historical value [5, 9, 21, 23, 35, 41, 46], or in aiding linguistic forensics experts in crime prevention or criminal investigation [7, 30, 39].

All of these tasks are usually approached as *text classification* tasks, whereby a supervised machine learning algorithm, using a set of labelled documents, is used to train a classifier to perform the required prediction task. As in many supervised learning endeavours, each training example is usually represented as a vector of features, where the value of a feature in a vector usually corresponds to the relative frequency with which a certain linguistic phenomenon (say, an exclamation mark, or a POS-gram) occurs within the document.

Koppel and Winter [29] describe an alternative method for generating vectorial representations of texts for authorship identification. Specifically, while in the standard representation methodology a vector represents a document, in this alternative method a vector represents an unordered *pair* of different documents. While in the standard methodology, the value of a feature is (an increasing function of) the relative frequency of occurrence of a given linguistic phenomenon in the document, in this alternative method, it is the absolute value of the *difference* between the relative frequencies (or increasing functions thereof) of this phenomenon in the two documents. Since these vectors represent differences, we call these representations *Diff-Vectors* (DVs). While in the standard methodology, the class label is the author of the document, in this DV-based methodology the class label is one of the two classes Same or Different (standing for "same author" or "different authors", respectively).

However, the goal of Koppel and Winter [29] was actually to propose a different method (the "impostors" method for SAV), and not to propose the DV-based methodology, which they dismiss as a "simplistic baseline method" [29, p. 179]. Since then, the use of DVs has never been studied systematically; to carry out such a systematic study is the goal of the present article.

The contributions of this article are thus as follows.

First, we study the consequences of the fact that, given n labelled documents, while the standard methodology gives rise to n training vectors, the DV-based methodology gives rise to $O(n^2)$ training vectors, which seems, at first sight, advantageous. Is this advantage for real? The present study answers this question.

Second, we carry out extensive experiments on a number of publicly available datasets (including one that we here make available for the first time) representative of different textual genres, lengths, and styles, with the goal of determining whether using DVs in place of “standard” vectors brings about higher accuracy in authorship identification tasks. In these experiments we tackle different authorship identification tasks, including SAV (for which DVs are naturally geared), AA, and AV; for these two latter tasks, we propose two new methods, *Lazy AA* and *Stacked AA* (two AA methods that can also be used for AV) that solve AA by using a DV-based SAV classifier as a building block. Our experiments show that the DV-based representation is advantageous, since it brings about substantially increased effectiveness. The experiments also show that DVs bring about substantial improvements especially in low-resource authorship analysis tasks, i.e., in tasks characterised by small quantities of training data (which is the case in many real-life authorship analysis scenarios, such as those dealing with ancient texts). Like the standard representation, the DV-based representation is learner-independent, i.e., it can be used in connection with any (supervised or unsupervised) learning method.

Third, we carry out an extensive comparative analysis of the efficiency of the two methodologies, both by studying the computational complexity of authorship analysis tasks and by clocking actual experiments. This study confirms that, as expected, the DV-based methodology is computationally more expensive; however, as we argue in detail, the additional computational cost is tolerable, especially in the light of the fact that, in authorship analysis, practical application scenarios often involve *a single* document of uncertain paternity, which means that classification efficiency is not a primary concern.

The rest of the article is structured as follows. In Section 2, we formally describe DVs and justify why they look like a superior means of representing authorship-related information; in particular, we show that DVs result in more training examples for the AV task (Section 2.3) and that DVs make training more robust in closed-set AA (Section 2.4). In Section 3, we describe algorithms for casting authorship identification tasks (such as AV or AA) in terms of SAV (the task that DVs are naturally designed for). Section 4 reports the results of our experiments; in particular, Section 4.4 discusses our “intrinsic” evaluation of DVs, i.e., one in terms of SAV, while Section 4.5 discusses an “extrinsic” evaluation of DVs, i.e., one in terms of downstream tasks such as AV and AA. Section 5 discusses related work, while Section 6 wraps up, also pointing at avenues for further research.

2 DIFF-VECTORS FOR AUTHORSHIP IDENTIFICATION

2.1 Authorship Identification Tasks

We assume a finite set $\mathcal{A}$ of authors (where $\mathcal{A}$ will be often called the *codeframe*) and a domain $\mathcal{D}$ of documents. For each document $x_i \in \mathcal{D}$, we indicate by $y_i \in \mathcal{A}$ the true author of x_i. We also assume the existence of a training set $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{n}$ of documents of known paternity.

We define AV as the task of predicting, given a document x_i and a candidate author $A^* \in \mathcal{A} = \{A_1, \ldots, A_m\}$, whether A^* is the author of x_i or not, where the labels $y_1, \ldots, y_n$ of the training documents are in $\mathcal{A} = \{A_1, \ldots, A_m\}$, with $m \geq 2$.[2]

We define *(closed-set)* AA as the task of predicting, given a document x_i and m candidate authors $\mathcal{A} = \{A_1, \ldots, A_m\}$, (one of whom is assumed to be the author of x_i), who among the members of $\mathcal{A}$ is the author of x_i, where the labels of the training documents are in $\mathcal{A} = \{A_1, \ldots, A_m\}$, with $m \geq 2$.[3]

[2] Note that, at training time, we assume to know the paternity (i.e., the labels) of documents written by authors other than A^*. Alternatively, AV can be formulated as a problem in which $m = 2$ and $\mathcal{A} = \{A^*, \overline{A^*}\}$, in which class $\overline{A^*}$ collectively represents the production of authors other than A^*. This special case will be discussed more in detail in Section 4.7.

[3] In real cases, we may not be certain that the author of x_i is indeed in $\mathcal{A} = \{A_1, \ldots, A_m\}$; in these cases, closed-set AA amounts to indicating who, among the authors in $\mathcal{A} = \{A_1, \ldots, A_m\}$, is the *most likely* author of x_i.

We define SAV as the task of predicting, given two unlabelled documents x_i and x_j, if they are by the same author or not, where the labels of the training documents are in $\mathcal{A} = \{A_1, \ldots, A_m\}$, with $m \geq 2$. This task admits two different variants, i.e., (i) *closed-set* SAV, which corresponds to the setup in which the authors of the unlabelled documents are assumed to be in $\mathcal{A}$, and (ii) *open-set* SAV, where the authors of the unlabelled documents are not necessarily in $\mathcal{A}$.

Note that terminology is somehow variable across the authorship analysis literature, and some of the above tasks may be defined slightly differently in other works. For instance, AV is sometimes defined (see e.g., [22]) as the task of predicting whether, given a document x_i and one or more documents known to be by a candidate author A^*, also x_i is by A^*. In this latter definition authorship verification shares some characteristics with "our" AV (in the fact that a candidate author A^* for document x_i is considered) and with "our" SAV (in the fact that we check whether x_i is by A^* by testing if x_i is by the same author as other texts known to be by A^*). Our definition of AV and SAV are, we think, cleaner, since they clearly separate (i) the task of predicting whether a document x_i is by a candidate author A^*, from (ii) the task of predicting whether a document x_i is by the same author as some other document. Our definitions are also more general, since "our" SAV does not assume the author of one of the two documents to be known.

2.2 Diff-Vectors

In "standard" authorship identification, each document x_i is represented via a labelled vector $\mathbf{x}_i$ of features, where each feature usually represents a linguistic phenomenon that may occur (possibly several times) in a document of $\mathcal{D}$, the label $y_i \in \mathcal{A}$ represents the true author of x_i, and the value $\mathbf{x}_i^k$ of the kth feature in vector $\mathbf{x}_i$ represents a non-decreasing function (e.g., tfidf) of the relative frequency of the linguistic phenomenon in x_i. For instance, if the kth feature stands for character 3-gram "car", then the value of $\mathbf{x}_i^k$ may be the number of occurrences of character 3-gram "car" in x_i divided by the number of all character 3-grams that x_i contains.

We here study an alternative type of vectorial representation for authorship identification tasks. Here, a labelled vector $\mathbf{x}_{ij}$ represents an *unordered pair* (x_i, x_j) of documents in $\mathcal{D}$ such that $i \neq j$, each feature represents a linguistic phenomenon that may occur (possibly several times) in a document of $\mathcal{D}$, the label $y_{ij} \in \mathcal{P} = \{\text{Same}, \text{Different}\}$ indicates whether the true authors of x_i and x_j are the same person or not, and the value $\mathbf{x}_{ij}^k$ of the kth feature in vector $\mathbf{x}_{ij}$ represents the absolute difference between non-decreasing functions of the relative frequencies of the linguistic phenomenon in x_i and x_j (In this section, we provisionally assume this function to be the identity function $f(x) = x$, while in the sections to come this function will be some well-established feature weighting function.) Since the *difference* between relative frequencies is central to the definition of these vectors, we call them *Diff-Vectors* (DVs).

If we have chosen our features well, i.e., if the frequencies of occurrence of the corresponding linguistic phenomena are indeed indicative of authorship, when two documents have been written by the same author the values $\mathbf{x}_{ij}^k$ of these features will be low, since the above frequencies will be similar in the two documents. In other words, DVs belonging to class Same will tend to be characterised by low feature values and low norms, while vectors belonging to class Different will tend to be characterised by high feature values and high norms. The quintessential (although fairly improbable) example of a DV likely to be in class Same is the vector of all 0's, since the fact that for all features, the frequency of occurrence of the feature in the two documents is identical, is highly indicative of the fact that (if the features have been chosen well) the two authors are the same person. Conversely, the quintessential (although fairly improbable) example of a DV likely to be in class Different is (if feature values are all normalised) a vector of all 1's, since it represents two documents with maximally different frequencies of occurrences for all features. All DVs fall, if normalised, in the unit hypercube.

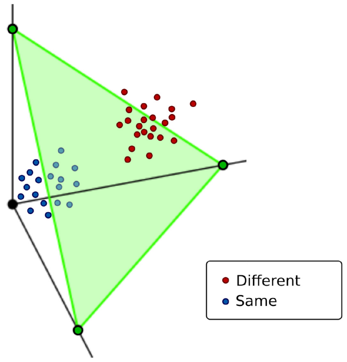

Fig. 1. 3-dimensional example of the surface (in green) that (ideally) separates the region of DVs belonging to Same (which corresponds to the tetrahedron comprised between the separating surface and the origin of the axes) and the region of DVs belonging to Different, in the linear case. When the number of features is t, the tetrahedron becomes a t-simplex and the separating surface is a $(t-1)$-simplex.

More in general, if a DV belongs to class Same, DVs that lie between it and the vector of all 0's will also tend (if we have chosen our features well) to belong to Same. As a result, the region that contains the DVs belonging to Same will tend to be the portion falling in the non-negative orthant of a star-convex region centred at the origin of the axis.[4] In particular, if Same and Different are linearly separable, and if t is the dimensionality of the feature space, the region that contains all the DVs belonging to Same will tend to be (see Figure 1) a t-simplex (in $t = 3$ dimensions: a tetrahedron) with an orthogonal corner, and the separating surface will tend to be a $(t-1)$-simplex (in $t = 3$ dimensions: a triangle).[5]

Any set of labelled documents $\mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ can be represented either in the standard way or via DVs. One of the main differences between the two representations is that the "standard" representation gives rise to n labelled vectors, while the alternative representation gives rise to $n(n-1)/2$ labelled vectors. The other main difference is that a classifier using the "standard" representation attempts to predict, given an unlabelled document, its true author, while a classifier using the DV-based representation attempts to predict, given two unlabelled documents, whether the two documents are or not by the same author. *In other words, the standard representation is geared towards AV or AA, while the DV-based representation is geared towards SAV.* However, AV and AA can (as discussed below) be recast in terms of SAV, and vice-versa; as a result, we will consider the two representations as general-purpose alternatives, and we will study them as such.

2.3 Diff-Vectors Result in More Training Examples for AV

Our working hypothesis is that the DV-based representation is advantageous. In order to show this, let us consider AV, and let us assume that $A^* \in \mathcal{A}$ is our candidate author. When using the standard representation, we typically replace each label in $\mathcal{A} \setminus \{A^*\}$ with label $\overline{A^*}$ (to indicate the complement of A^*) and train a binary classifier that discriminates between A^* and $\overline{A^*}$. However, in doing so a lot of information is lost, namely, the information whether two training examples in $\overline{A^*}$

[4] The *non-negative orthant* is the generalisation to $t > 2$ dimensions of the 1st quadrant of the familiar 2-dimensional Cartesian space. A *star-convex region* centred at point $\mathbf{x}_0$ is a region of t-dimensional space in which for every point $\mathbf{x}$ in the region, all points between $\mathbf{x}_0$ and $\mathbf{x}$ are also in the region.

[5] A *t-simplex* is the generalisation to $t > 3$ dimensions of the 2-dimensional notion of triangle and the 3-dimensional notion of tetrahedron. A t-simplex with an *orthogonal corner* is one that has a vertex such that all its adjacent edges are pairwise orthogonal.

are by the same author or not. For authorship-related tasks, this is valuable information, which the standard representation wastes and the DV-based representation does not. The following example shows that the information wasted by the standard representation is, indeed, *a lot*.

Example 2.1. Assume a set of 10 authors and a training set consisting of 100 training examples for each author. The DV-based representation gives rise to (1,000·999)/2 = 499,500 DVs, among which:

(1) 10·(100·99)/2 = 49,500 examples have label Same, since for each author $A_z \in \{A_1, \ldots, A_{10}\}$ there are (100·99)/2 = 4,950 unordered pairs of different examples such that the author of both examples is A_z; of these
  (a) (100·99)/2 = 4,950 are such that the author of both examples is A^*;
  (b) 9·(100·99)/2 = 44,550 are such that the author of both examples is A_z for some $A_z \neq A^*$;
(2) 45·(100·100) = 450,000 examples have label Different, since there are 10·9/2 = 45 unordered pairs (A', A'') of different authors, and for each such pair there are 100·100 = 10,000 pairs of examples in which one example is by A' and the other example is by A''; of these
  (a) 9·(100·100) = 90,000 are such that one of A' and A'' is A^*;
  (b) 36·(100·100) = 360,000 are such that neither of A' and A'' is A^*.

Note that the information provided to the training process by the examples of Type 1a is also provided (albeit in a different form) when using the standard representation, since with the latter the learner is implicitly told that the two documents are from the same author. The same happens for the examples of Type 2a, since with the standard representation, the learner is implicitly told that the two documents are from different authors. However, the key observation here is that the examples of Type 1b and Type 2b provide information that is instead lost when using the standard representation, since the standard representation only tells the learner that the two documents are not by A^*, but does not tell the learner if they are by the same author or not. In sum, 404,550 out of 499,500 training examples, i.e., about 81% of the entire set, provide information that was not provided by the standard representation; in other words, in this case, the learner receives more than 5 times the amount of information than the standard representation provides to it.

More in general, if we have m authors and $q = n/m$ training examples per author, the number of DVs that do *not* provide additional information with respect to the standard representation is

$$\frac{q(q-1)}{2} + (m-1)q^2 \tag{1}$$

i.e., the number of pairs of Type 1a plus the number of pairs of Type 2a, while the number of DVs that do provide additional information is

$$\frac{(m-1)q(q-1)}{2} + \frac{(m-1)(m-2)q^2}{2} \tag{2}$$

i.e., the number of pairs of Type 1b plus the number of pairs of Type 2b. Note that, while the amount of information that was already available to the learning process is $O(mq^2)$ (Equation (1)), the new information made available to it is $O(m^2q^2)$ (Equation (2)). The latter amount of information can be extremely valuable, especially since it comes at no cost, and especially in application scenarios (as there are many in authorship identification) characterised by the scarcity of training data. Among all of the above,

$$\frac{q(q-1)}{2} + \frac{(m-1)q(q-1)}{2} = \frac{mq(q-1)}{2} \tag{3}$$

are examples of Same, which are $O(mq^2)$, while

$$(m-1)q^2 + \frac{(m-1)(m-2)q^2}{2} = \frac{m(m-1)q^2}{2} \tag{4}$$

are examples of Different, which are $O(m^2q^2)$.

In sum, when our task is AV, if we switch from standard representations to DV-based representations, we end up with a much higher quantity of training data, since DV-based representations exploit information that standard representations waste. However, note that switching from standard representations to DV-based representations means switching (as noted at the end of Section 2.2) from vectors geared towards AV to vectors geared towards SAV. This suggests the idea to use these vectors to train a high-performance SAV classifier, and then to devise an algorithm that can perform AV on top of this SAV classifier; this is the goal we will pursue in Section 3.3.

2.4 Diff-Vectors Make Training More Robust in Closed-Set AA

The fact that more information is provided to the training process holds for AV, but does not necessarily hold for other authorship identification tasks. In general, this fact only holds for tasks in which, as in AV, the training documents by different authors end up being grouped together into a single class; this happened in AV with the $\overline{A^*}$ class. However, that more information is provided to the training process does not hold when the above-mentioned grouping does not happen, as e.g., in closed-set AA. In the latter task, the information conveyed to the training process by a DV with label Same is obviously also implicitly conveyed when using the standard representation (where the vectors corresponding to the two documents are labelled with the same author), and the same holds for DVs with label Different (where the two vectors are labelled with different authors).

So, in closed-set AA (and in the latter tasks in general) it would appear that there is no advantage in using DVs. This is actually not true because the advantage is in the fact that, *when using DVs, all the training information is concentrated on labelling just two classes*, i.e., Same and Different, while in the classical representation, this information is spread out thin, i.e., it is used for labelling m different classes, each of which thus ends up having a smaller number of positive training examples. The following example makes the point more concrete.

Example 2.2. Assume we are dealing with closed-set AA; assume a set of $m = 10$ authors and a training set consisting of $q = 20$ training examples for each author. The standard representation gives rise to $q \cdot m = 200$ training vectors, 20 for each class, while the DV-based representation gives rise to $mq(mq-1)/2 = 19{,}900$ training vectors, among which $10{\cdot}(20{\cdot}19)/2 = 1{,}900$ DVs for class Same and $10 \cdot 9 \cdot 20^2/2 = 18{,}000$ DVs for class Different.

More in general, if we have m authors and q training examples per author, in closed-set AA we have $mq(q-1)/2$ DVs of class Same and $m(m-1)q^2/2$ DVs of class Different, which means that the ratio between the number of training examples of Same and the number of training examples of Different is

$$\frac{q-1}{q(m-1)} \approx \frac{1}{m-1}$$

This indicates that we are in the presence of an *imbalanced* binary classification problem (which is even more imbalanced if m is large); however, this is not a problem because, since we typically have many training DVs (see e.g., Example 2.2), we can subsample class Different, i.e., remove some among its many training examples from the training set.

In sum, the use of the DV-based representation in closed-set AA allows the SAV binary classifier to be trained robustly, thanks to the fact that the existing amount of training information can be devoted to solving a comparatively easier binary classification task rather than a comparatively more difficult 1-of-m classification task. We can thus expect to obtain accurate SAV classification predictions; in Section 3.2, we will see that these SAV predictions can also be used by a downstream process to solve authorship identification tasks such as AV and AA.

3 SOLVING SAV, AA, AND AV, BY MEANS OF DIFF-VECTORS

One difference between the standard representation, in which class labels represent authors, and the representation based on DVs, in which class labels are in {Same, Different}, is that the tasks that can be solved "directly" are AV and AA for the former, and SAV for the latter. That is, by using the standard representation, AV and AA can be solved directly by setting up a classifier that, for a given document, returns a class label in $\mathcal{A}$ (for AA) or in $\{A^*, \overline{A}^*\}$ (for AV); SAV is instead to be solved as a derivative, "downstream" task, e.g., by first determining the true authors of documents x_i and x_j by means of two calls to an AA engine, and then checking whether the two returned class labels are the same or not.[6] On the contrary, when using the DV-based representation, SAV is solved directly; AV and AA are instead to be solved as derivative tasks, using SAV as the building block of any algorithm for solving them. In this section, we first formally define our method for performing SAV (Section 3.1), and then go on to describe two alternative solutions for solving both AV and AA (Section 3.2) that build on top of the former.

3.1 Solving SAV by Means of Diff-Vectors

Given a training set $\mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of documents $x_i \in \mathcal{D}$ labelled by classes $y_i \in \mathcal{A} = \{A_1, \ldots, A_m\}$ representing authors, we define its *pair-based version* as

$$\mathcal{L}_{\mathcal{P}} = \{((x_i, x_j), \text{SD}(y_i, y_j)) \mid i, j \in \{1, ..., n\}, j < i\} \tag{5}$$

where $\text{SD}(y_i, y_j)$ is an indicator function that returns Same if $y_i = y_j$ and Different otherwise. We also assume a feature extractor $f : \mathcal{D} \rightarrow \mathbb{R}^t$ which maps documents $x \in \mathcal{D}$ into t-dimensional vectors $\mathbf{x}$ of real numbers. We can thus rewrite $\mathcal{L}$ as $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and redefine $\mathcal{L}_{\mathcal{P}}$ as

$$\mathcal{L}_{\mathcal{P}} = \{(\mathbf{x}_{ij}, \text{SD}(y_i, y_j)) \mid i, j \in \{1, \ldots, n\}, j < i\} \tag{6}$$

where $\mathbf{x}_{ij} \in \mathbb{R}^t$ is a vector of absolute differences of feature values, i.e., $\mathbf{x}_{ij}$ is the vector such that its kth component is $\mathbf{x}_{ij}^k = |\mathbf{x}_i^k - \mathbf{x}_j^k|$, for all $1 \leq k \leq t$, $i \neq j$.

Note that $|\mathcal{L}| = n$ while $|\mathcal{L}_{\mathcal{P}}| = n(n-1)/2$, i.e., the pair-based version $\mathcal{L}_{\mathcal{P}}$ is $(n-1)/2$ times larger than its standard counterpart $\mathcal{L}$. In practice, the size of $\mathcal{L}_{\mathcal{P}}$ can be so large as to make the learning process intractable for some batch learners. For example, the 499,500 training DVs of Example 2.1 would result from a dataset of 10 authors and 100 training documents per author, which is not a terribly large dataset. As shown in Section 2.4, $\mathcal{L}_{\mathcal{P}}$ tends to be imbalanced, with a Same/Different training example ratio close, assuming a training set containing the same number of documents for each author, to $1/m$.

In practice, we will be interested in generating and using only a subset $\mathcal{L}'_{\mathcal{P}} \subset \mathcal{L}_{\mathcal{P}}$: by including in $\mathcal{L}'_{\mathcal{P}}$ a small enough number of elements of $\mathcal{L}_{\mathcal{P}}$, we can make the training process tractable, and by including in $\mathcal{L}'_{\mathcal{P}}$ an equal number of examples of Same and Different we can avoid the typical negative consequences of imbalance. By using a subset $\mathcal{L}'_{\mathcal{P}}$ with these characteristics, we can then train a binary classifier $h : \mathbb{R}^t \rightarrow \{\text{Same}, \text{Different}\}$. We can this classifier DV-Bin, since it is a binary classifier that uses DVs.

Without loss of generality, and for ease of notation, we will henceforth use h as the function of two arguments $h : \mathcal{D} \times \mathcal{D} \rightarrow \{\text{Same}, \text{Different}\}$, thus leaving implicit the phases of

[6] This is possible only for closed-set SAV, though, since open-set SAV cannot be recast in terms of AA.

(a) mapping documents to feature vectors, and (b) computing DVs from the absolute differences of feature values. As a result, we can simply write $h(x_i, x_j)$ to indicate a predicted label in {Same, Different}.

3.2 Solving AA by Means of Diff-Vectors

In this section, we describe how SAV can be used to implement AA as downstream tasks.

In order to predict by whom among the authors in $\mathcal{A}$ a test document x has been written, and to do so by using a SAV classifier, it makes sense to look at how x relates to the training documents in terms of the Same and Different classes. For instance, if for all documents $x' \in \mathcal{L}$ written by A_z the pair (x, x') is assigned by the SAV classifier to class Same, and if for all $x'' \in \mathcal{L}$ written by an author in $\mathcal{A} \setminus \{A_z\}$ the pair (x, x'') is assigned to class Different, it would be reasonable to predict that x has been written by A_z.

Unfortunately, this uniformity rarely occurs in practice: in more typical cases, the SAV classifier will assign to class Same, say, some pairs (x, x') where x' has been written by A_z, and some pairs (x, x'') where x'' has been written by an author other than A_z. This brings up the question: how should we act in the presence of such apparently contradictory outcomes?

Given that we need to build our AA algorithm on top of the output of the SAV classifier, it is in our best interest to squeeze every possible bit of information from this output. As a result, we will be interested in exploiting not just the binary prediction of the SAV classifier, but also its non-binary classification score, representing the degree of certainty with which it has issued this prediction. We assume that our SAV classifier is of the form

$$h : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1] \tag{7}$$

i.e., returns classification scores that are *posterior probabilities*. These latter are values $\Pr(\text{Same}|x_i, x_j)$ that denote the probability that the SAV classifier attributes to the fact that x_i and x_j have been written by the same author, and are such that $\Pr(\text{Different}|x_i, x_j) = 1 - \Pr(\text{Same}|x_i, x_j)$.

We explore two techniques for building AA classifiers on top of SAV classifiers, one inspired by *lazy learning* methods [1] and another inspired by the well known *Stacked Generalisation* algorithm [49].

3.2.1 Lazy AA. The first SAV-based AA algorithm that we explore in this article, and that we call *Lazy AA*, draws inspiration from distance-weighted k-NN, but is different from it. Similarly to distance-weighted k-NN, the underlying idea of our method is that, given a test document x, if a training document x' authored by A_z is "stylistically similar" to x, this brings evidence towards the fact that also x is authored by A_z, and this evidence can be quantified exactly by the amount of stylistic similarity. Differently from distance-weighted k-NN, though, instead of having access to a function that computes the similarity between two documents, we here have access to a SAV (soft) classifier that computes the probability that the two documents are in class Same. It is thus just natural to compute the stylistic similarity between x and x' as $\Pr(\text{Same}|x, x')$, i.e., as the probability that the SAV classifier attributes to the fact that x and x' have been written by the same author.

Our combination rule thus consists of selecting, for each author $A_z \in \mathcal{A}$, the k training documents written by A_z that are stylistically most similar to our test document x (i.e., the ones for which $\Pr(\text{Same}|x, x')$ is highest), and computing the average value of this stylistic similarity across these k documents; the author for which this average stylistic similarity is highest is predicted to be the author of x. In symbols, this comes down to

$$\begin{aligned} h'(x, \mathcal{L}, k) &= \underset{A_z \in \mathcal{A}}{\arg\max} \frac{1}{k} \sum_{x_i \in \mathrm{NN}(k, \mathcal{L}, A_z, x, h)} h(x, x_i) \\ &= \underset{A_z \in \mathcal{A}}{\arg\max} \frac{1}{k} \sum_{x_i \in \mathrm{NN}(k, \mathcal{L}, A_z, x, h)} \Pr(\mathrm{Same}|x, x_i) \end{aligned} \tag{8}$$

where $\mathrm{NN}(k, \mathcal{L}, A_z, x, h)$ returns the k documents from training set $\mathcal{L}$ that have been written by author A_z and are closest to x according to the SAV classifier h. Note that the h' functional is parameterised by $\mathcal{L}$ (and k) since, as in all lazy learning methods, there is no proper training phase for h', and all the computation is carried out at classification time.

The optimal value for parameter k can be found via ‘‘leave-one-out” (LOO) validation on the training set $\mathcal{L}$. That is, for each value of k in the tested range each training document $x_i \in \mathcal{L}$ is classified by a classifier h' trained on $\mathcal{L} \setminus \{x_i\}$; k is thus set to the value that maximises a given effectiveness measure as computed on the entire set $\mathcal{L}$.[7] If we use *(vanilla) accuracy* (i.e., the proportion of correctly classified instances) as the effectiveness measure, this process comes down to computing

$$k^* = \arg\max_k \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{L}} \mathbf{1}[h'(x_i, \mathcal{L} \setminus \{x_i\}, k) = y_i] \tag{9}$$

where $\mathbf{1}[s]$ is an indicator function returning 1 if statement s is true and 0 otherwise. This optimisation can be performed very quickly if the posterior probabilities $\Pr(\mathrm{Same}|x_i, x_j)$ are computed only once for all $x_i, x_j \in \mathcal{L}$ and stored for fast reuse. Similarly, $\mathrm{NN}(k, \mathcal{L}, A_z, x, h)$ can be made to return the top k elements (for different values of k) from a fully ranked list that is computed once and reused when necessary. Note also that the majority of these operations are amenable to parallelisation.

3.2.2 *Stacked AA.*

Stacked AA (so called since it is inspired by *stacked generalisation*—[49]) consists of an AA (single-label multiclass) classifier h', trained by general-purpose learning algorithms, that classifies documents represented by vectors of posterior probabilities $\Pr(\mathrm{Same}|x, x_k)$, each of which has been returned by an underlying, previously trained SAV classifier h (more precisely, a DV-Bin classifier of the type described in Section 3.1). More in detail, in order to predict who among the authors in $\mathcal{A}$ has written document x, we represent x via a vector

$$\begin{aligned} \phi(x) &= (h(x, x_1), \ldots, h(x, x_n)) \\ &= (\Pr(\mathrm{Same}|x, x_1), \ldots, \Pr(\mathrm{Same}|x, x_n)) \end{aligned} \tag{10}$$

of n posterior probabilities, one for each training example in $\mathcal{L}$. The kth value in this vector is the value $h(x, x_k) = \Pr(\mathrm{Same}|x, x_k)$, where x_k is the kth training example. In other words, in order to classify x we first need to perform $|\mathcal{L}|$ SAV classifications, where the kth such classification attempts to predict whether the test document x was written by the same author who also wrote training document x_k.

[7] One might wonder why we go for LOO, a traditionally expensive (and sometimes *too* expensive) way of optimising parameters, rather than the cheaper t-fold cross-validation (t-FCV). The reason is that, in our case, LOO is no more expensive than t-FCV because we are in a *lazy* learning context. In other words, in traditional *eager* learning contexts LOO requires $|\mathcal{L}|$ classifier retrainings, while t-FCV requires only $t \ll |\mathcal{L}|$ classifier retrainings; however, in lazy learning contexts there are no retrainings because classifiers are not “trained”, since all inductive inference is carried out at classification time.

At training time, we train the AA classifier h' by using all the training examples in $\mathcal{L}$ represented in the style of Equation (10). In other words, by applying the mapping $\phi : \mathbb{R}^t \rightarrow [0,1]^n$ to the training documents themselves we define a new "view" $\mathcal{L}_h = \{(\phi(x_i), y_i)\}_{i=1}^n$ of the training set $\mathcal{L}$, in which the training documents are not represented via vectors of t stylometric features but, thanks to the underlying SAV classifier, via vectors of $|\mathcal{L}|$ posterior probabilities, with $\phi(x) \in [0,1]^n$. The training set $\mathcal{L}_h$ can directly be used to train a general-purpose classifier $h' : [0,1]^n \rightarrow \mathcal{A}$ in the feature space of posterior probabilities.[8] Of course, in order to generate $\mathcal{L}_h = \{(\phi(x_i), y_i)\}_{i=1}^n$ we first need to train a SAV classifier h via the DV-Bin method of Section 3.1. In the experiments of Section 4, we will concentrate on instantiations of h' that are generated by the same learning method (e.g., logistic regression) used to generate h.

At classification time, a given test document x is classified by first computing $\phi(x)$ (this requires invoking n times classifier h) and then invoking classifier $h'(\phi(x))$.

There are several important aspects in which Stacked AA differs from Lazy AA:

- in Stacked AA, evidence is provided by *all* training examples, and not just by the k examples most similar to the test example, as is instead the case in Lazy AA;
- in Stacked AA, the combination rule (i.e., the rule that assembles the evidence provided by the training examples into a final decision) is *learnt* by a metaclassifier, i.e., it is not static, as is instead the case in Lazy AA;
- in Stacked AA, learning is performed offline (since the metaclassifier is trained before the testing phase begins), while in Lazy AA all inductive inference is carried out at classification time.

One important aspect in which Stacked AA differs from Stacked Generalisation, instead, is that in Stacked Generalisation the metaclassifier and the base classifiers are *homogeneous*, i.e., all use the same set of classes, while in Stacked AA, the metaclassifier and the base classifiers are *heterogeneous*, i.e., use different sets of classes. Indeed, the base classifiers use the classes in {Same, Different}, since they are binary SAV classifiers, while the metaclassifier use the classes in $\mathcal{A} = \{A_1, \ldots, A_n\}$, since it is a single-label multiclass AA classifier.

3.3 Solving AV by Means of Diff-Vectors

It is fairly straightforward to take the algorithms described in Sections 3.2.1 and 3.2.2 and generate versions (that we will dub *Lazy AV* and *Stacked AV*) that solve AV instead of AA. The only difference between Lazy AV and Lazy AA, and between Stacked AV and Stacked AA, is that in the AV versions of the two algorithms the codeframe used is binary, i.e., it is $\mathcal{A} = \{A^*, \overline{A^*}\}$; in particular, this means that for Stacked AV the metaclassifier h' is a binary classifier instead of a multiclass classifier. Everything else is unmodified.

However, in preliminary experiments that we have run, both Lazy AV and Stacked AV proved substantially inferior to versions of Lazy AA and Stacked AA, respectively, in which we attribute document x to A^* if the AA algorithm does so and we attribute x to $\overline{A^*}$ if the AA algorithm attributes it to an author A_z different from A. Concerning the reason why Lazy AV underperforms Lazy AA, this has likely to do with the fact that there is an *a priori* high probability that the k nearest neighbours in $\overline{A^*}$ are, on average, closer to x than the k nearest neighbours in A^*, since $\overline{A^*}$ is a very large pool to choose from (this does not happen in AA, where, assuming an equal number of training documents per author, all pools are equally large); this can give undue advantage to $\overline{A^*}$

[8] Note that, if the learning algorithm is a linear model, then it takes the form of $h'(x) = \sum_{l=1}^n \alpha_l h(x, x_l)$, in which $\{\alpha_l\}_{l=1}^n$ are the parameters to be learned, and the set of functions $\{h(\cdot, x_l)\}_{l=1}^n$ plays the role of a set of basis functions centred at the training points.

over A^*, and thus generate a large quantity of false negatives. Concerning the reason why Stacked AV underperforms Stacked AA, this has likely to do with the fact that the metaclassifier of Stacked AV does not put the available class information to the best use, i.e., conflates all labels different from A^* into a single label $\overline{A^*}$ that ends up being poorly characterised from the semantic point of view.

Therefore, in the rest of the article, the algorithms we will use for solving AV via DV-based representations will be the versions of Lazy AA and Stacked AA described at the beginning of the previous paragraph. A consequence of this is that any AA experiment that involves the use of either Lazy AA and Stacked AA and a codeframe $\mathcal{A} = \{A_1, \dots, A_m\}$, is also *de facto* a set of m different AV experiments. In other words, we will not need to run separate AA and AV experiments, i.e., we will evaluate the AA experiments that we describe in Section 4 both in terms of AA *and* AV.

4 EXPERIMENTS

In order to test whether a representation based on DVs is advantageous with respect to a representation based on standard vectors, we compare these two different design choices in experiments that we run on four publicly available datasets (among which one that we here make available for the first time) and for all three authorship analysis tasks (AA, AV, SAV). The code to reproduce our experiments is available online at https://github.com/AlexMoreo/diff-vectors.

4.1 Datasets

We run experiments on four datasets consisting of textual documents annotated by author; our datasets are representative of different textual genres, lengths, and styles, are publicly available, and all consist of English texts. The four datasets are:

- IMDB62. This dataset[9] was created and made publicly available (along with an extended version, IMDB1million) by Seroussi et al. [42]. It contains film reviews collected from the popular Internet Movie Database, and accounts for 62 authors/reviewers and 1,000 reviews authored by each of them. In order to divide the 62,000 documents into a training set and a test set, we perform a stratified split, resulting in 700 training documents and 300 test documents for each author. We use these texts as examples of a "moderately formal" type of communication, since the reviews are not as short as, for example, online messages, and, despite some occasional slang, are written in a clear and correct (although often informal) manner.
- PAN2011. This dataset[10] was created for the PAN 2011 international authorship identification competition [2]. The dataset is based on the Enron e-mail corpus [26], i.e., the documents are e-mails annotated by author. Klimt and Yang [26] have removed personal names and e-mail addresses and replaced them with specific tags, which means that an authorship identification method is not able to use this extremely revealing information. In our experiments, we use the "Large" versions of the training and test sets. We discard documents containing fewer than 15 words, thus ending up with 7,111 training documents and 1,157 test documents, altogether accounting for 70 different authors. The e-mails are often extremely short, and show many characteristics of online communication; in order to avoid texts which are excessively short (and thus too difficult to attribute), we remove e-mails consisting of fewer than 15 words.

[9] Available at: https://umlt.infotech.monash.edu/?page_id=266

[10] Available at: https://pan.webis.de/clef11/pan11-web/authorship-attribution.html

- Victorian. This dataset[11] was created and made publicly available by Gungor [17]. It consists of books by American or English 18th–19th century novelists, subdivided into segments of 1,000 words each by the creator of the dataset, who also (i) removed the first and last 500 words of each book and, (ii) as a topic-filtering measure, retained only the occurrences of the 10,000 words most frequent in the dataset. The result is a corpus of more than 50,000 documents (i.e., segments) by 50 different authors. We restrict our attention to the "training" partition made available by the authors, which accounts for 45 different authors. The corpus is an imbalanced one, with the least represented author accounting for 183 segments and the most represented one accounting for about 6,914 of them. In order to divide it into a training set and a test set, we perform again a stratified split, including 70% of each author's texts in the training set and the remaining 30% in the test set. We use these documents as examples of literary production characterised by a sophisticated style.
- arXiv. This dataset, which we have created and made publicly available ourselves,[12] consists of abstracts of single-author papers from arXiv.[13] In order to limit domain-dependence we have harvested these abstracts by querying arXiv's API with a list of computer-science-related keywords, mostly focused on machine learning.[14] Computer science articles are seldom written by a single author, which means that this dataset is not large. The corpus somehow follows a power-law distribution, with few prolific authors and many authors accounting for very few abstracts each: we retain authors with at least 10 abstracts to their name, resulting in a total of 1,469 documents from 100 authors. The 2 most prolific authors have 34 abstracts to their name, the 10 most prolific authors have written 22 or more, while 50% of the authors have no more than 12 abstracts to their name. In order to divide the corpus into a training set and a test set, we perform a stratified split, with the production of each author being split into a training set (70% of the abstracts) and a test set (30%). We use these abstracts as examples of "scientific communication", characterised by a precise and compact style, with an abundance of technical terminology.

Table 1 conveniently summarizes the main characteristics of these four datasets.

4.2 Learners

We use logistic regression (LR) as the learning method. LR is a simple linear model that has delivered very good accuracy in a number of text-related applications. LR has two further advantages, i.e., (i) the classification scores returned by the classifiers trained by it are posterior probabilities, and (ii) these probabilities are *well-calibrated.*[15] These are important advantages, since the

[11] Available at: https://archive.ics.uci.edu/ml/datasets/Victorian+Era+Authorship+Attribution

[12] Available at: https://doi.org/10.5281/zenodo.7404702

[13] https://arxiv.org/

[14] The query used was "deep learning, machine learning, information retrieval, computer science, data mining, support vector, logistic regression, artificial intelligence, supervised learning".

[15] A well-calibrated classifier is one that returns accurate posterior probabilities. An intuition of what "accurate posterior probabilities" means can be provided by the following example. If 10% (resp., 90%) of all the documents x_i for which $h(x_i) = \Pr(A|x_i) = 0.5$ indeed belong to class A, we can say that the classifier h has overestimated (resp., underestimated) the probability that these documents belong to A, and that their posteriors are thus inaccurate. Conversely, if this percentage is 50%, we can say that the classifier h has correctly estimated the probability that these documents belong to A, and that their posteriors are thus accurate. Indeed, we say (see, for instance, Reference [14]) that the posteriors $h(x_i) = \Pr(A|x_i)$ are perfectly calibrated (i.e., accurate) with respect to a (labelled) set $\sigma = \{(x_i, y_i)\}_{i=1}^{n}$ if, for all $\alpha \in [0, 1]$, it holds that

$$\frac{|\{(x_i, y_i) \in \sigma \mid h(x) = \alpha, y_i = A\}|}{|\{(x_i, y_i) \in \sigma \mid h(x) = \alpha\}|} = \alpha \tag{11}$$

Table 1. Main Characteristics of the four Datasets Used in this Work

Dataset	Proposed in	Type of docs	# of authors	Min # of docs per author	Max # of docs per author	Total # of docs	Average doc length (words)
IMDB62	[42]	film reviews	62	1,000	1,000	62,000	349.0
PAN2011	[2]	e-mails	70	1	561	8,268	69.2
Victorian	[17]	[segments of] novels	45	183	6,914	53,678	1,000.0
arXiv	[this work]	abstracts of papers	100	10	34	1,469	129.3

methods we have described in Sections 3.2.1 and 3.2.2 do rely on posterior probabilities, and obviously benefit from the fact that these posteriors are high-quality.

We optimise the hyperparameter C of LR (the inverse of the L2 regularisation strength) in the log-space $\{10^i\}_{i=0}^{i=4}$, and select the value of C that minimizes the multinomial loss in a stratified t-fold cross-validation (with $t = 5$).[16]

In order to generate the Same and Different training pairs, we adopt the following policy. Given a training set $\mathcal{L}$, we first compute the number of Same pairs that can be generated. If there are fewer than 50,000 Same pairs, we generate them all; otherwise, we draw (uniformly at random) and generate 50,000 Same pairs. We then draw (again, uniformly at random) and generate as many Different pairs as the Same pairs we have generated. This is in order to guarantee a balanced training set, since there are usually many more potential Different pairs than Same ones.

4.3 Features

As for the choice of features, we stick to ones well-known and broadly adopted in the field of authorship analysis, i.e., features of a frequentistic nature that can be extracted automatically and that are believed to convey stylistic information; see, for example [13, 20, 43] for an overview, and Kestemont et al. [24, 25] for a discussion of the most frequently used features in recent shared tasks focused on authorship analysis. These features are considered a standard in the authorship analysis field because they represent linguistic traits that are believed to remain more or less constant in an author's production and, conversely, to vary in noticeable fashion across different authors [20, p. 241]; as such, they tend to be identifiers of the idiosyncratic style of an author. Note that other

The classifiers trained by means of some learners (and LR is one of them) are known to return reasonably well-calibrated probabilities. Those trained by means of some other learners (such as Naïve Bayes) return probabilities which are known to be not well calibrated [12]. Yet other learners (such as SVMs or AdaBoost) train classifiers that return confidence scores that are not probabilities (i.e., that do not range on [0,1] and/or that do not sum up to 1). In order to address these two latter cases, *probability calibration* mechanisms exist (see e.g., [36–38, 50, 52]) that convert the outputs of these classifiers into well calibrated probabilities.

[16] We use the LogisticRegressionCV scikit-learn's implementation, see https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html

sets of features could have been equally plausible;[17] however, this is not an important concern for our work, since it is completely agnostic with respect to the specific features that should be used.

The features we use can be naturally subdivided into two groups. Group 1 is composed of

- *Function words.* Each function word that appears in the training set is a feature in our vectorial representations. We use the list of English function words provided by NLTK.[18]
- *Word lengths.* Each word length instantiated in the training set is a feature.
- *Sentence lengths.* Each sentence length instantiated in the training set is a feature.
- *Punctuation symbols.* Each punctuation symbol that occurs in the training set is a feature.

while Group 2 is composed of

- *POS n-grams.* We extract parts of speech from our texts by using the Spacy library,[19] and we consider each POS n-gram (for $n \in [3, 4]$) that occurs in the training set as a potential feature (where "potential" means "barring feature selection"—see below).
- *Word uni-grams.* We consider each word that occurs in the training set as a potential feature.
- *Character n-grams.* We consider each character n-gram (for $n \in [2, 5]$) that occurs in the training set as a potential feature.

The features in Group 1 (i) are relatively few (typically: $O(10^2)$), and (ii) are dense, i.e., all of them can be expected to occur to some degree in most texts. Given one of these features and given a document, as the value of the feature in the document, we take its relative frequency in the document; for instance, the value of punctuation symbol "!" in a document will be the number of times symbol "!" occurs in the document divided by the number of punctuation symbols in the document. We also apply standardisation to the columns that these features generate in the document-by-feature matrix.[20]

The features in Group 2 are many (typically: $O(10^4)$ or $O(10^5)$). In order to deal with the fact that they may be *too* many, we apply to them filter-style feature selection, using the chi-square test as the term scoring function [51] and retaining the 50,000 highest-scoring features. As the feature weighting function, rather than using plain relative frequency, we use tfidf (an increasing function of relative frequency) in its standard "ltc" variant (see e.g., [40]).[21] The features in Group 2 are sparse, i.e., in a given document, a large number of them will not occur in it; we do not apply any standardisation to the features in Group 2 since this would turn them into dense features, and this would be detrimental to efficiency.

4.4 Intrinsic Evaluation of Diff-Vectors

Our "intrinsic" evaluation of DVs consists of SAV experiments, since SAV is the task that a classifier using DVs can solve directly. In these experiments, we use a set of authors $\mathcal{A} = \{A_1, \dots, A_m\}$,

[17] For instance, in Reference [8], we use function words, word lengths, sentence lengths, and POS n-grams, but we do not use punctuation symbols (since in that work we deal with medieval Latin texts, and since Latin does not have punctuation), word-unigrams, and character n-grams; we instead use features based on "syllabic quantity", which are specific to Latin.

[18] https://www.nltk.org/

[19] https://spacy.io/

[20] Standardisation (aka z-scoring) is a normalisation process consisting of centring and scaling a random variable so as to force its distribution to have 0-mean and 1-variance, i.e., the z-score of a raw variable x is defined as $z = \frac{x-\mu}{\sigma}$ where μ and σ are the (sample) mean and (sample) standard deviation of x as estimated in the training set. For the benefits in accuracy deriving from standardising dense features, see Reference [33].

[21] Using tfidf (which is indeed an increasing function of relative frequency) for weighting sparse features is customary in authorship analysis (see e.g., References [19, 28, 29, 31]). This function is the combination of the tf factor, which is somehow akin to relative frequency, with the idf factor, which lends a higher weight to features that are rare in the training set (see [40] for details); in authorship analysis, the use of idf is justified by the fact that rare POS n-grams/word unigrams/character n-grams can be considered more indicative of style than common ones.

with $m > 2$, each one being the author of q documents. Given a dataset that contains a test set $\mathcal{U}$, we test our systems on randomly drawn samples of test document pairs. The reason why we do not test on *all* possible pairs is (see also Section 2.3) a practical one, i.e., the fact that the number $|\mathcal{U}|(|\mathcal{U}|-1)/2$ of all possible pairs is too high for all but the most trivial datasets. We randomly draw balanced subsets of 1,000 test pairs (500 positive and 500 negative) for each experiment.

We investigate the impact on performance of the number m of authors and the number q of training documents per author. Specifically, for the `IMDB62`, `PAN2011`, and `Victorian` datasets, we run experiments varying the number m of authors in the set $\{5, 10, 15, 20, 25\}$ and the number q of documents per author in the set $\{10, 20, 30, 40, 50\}$. Samplings are incremental, i.e., we do not resample from scratch; in other words, when moving from, say, $q = 20$ to $q = 30$, we add 10 new documents per author to the previous 20. Regarding the test set, for each choice of m we draw 2,000 random test pairs, 1,000 of which consist of texts written by some among the m authors present in the training set (*closed-set SAV*), while the other 1,000 pairs consist of texts written by m authors other than the m authors present in the training set (*open-set SAV*).[22]

In order to compensate for the random effect introduced by sampling (authors, documents, and test pairs), we report results obtained by averaging across 10 runs for each combination (dataset, m, q); we use the same random samples for all the methods we compare. The only exception is the `arXiv` dataset, which, due to its limited size, does not allow this extraction of multiple samples; hence, for this dataset, we simply report experiments across 10 random train/test splits of the entire dataset.

We perform experiments in both closed-set SAV (Section 4.4.1) and open-set SAV settings (Section 4.4.2). We evaluate the performance in terms of vanilla accuracy (fraction of correctly classified pairs), which is a perfectly valid evaluation measure when the test set is balanced across the classes, such as the present one.

4.4.1 Experiments on Closed-Set SAV. In the closed-set scenario, the authors in the test set $\mathcal{U}$ are the same as in the training set. We here explore two variants of our method:

- DV-Bin: the binary classifier discussed in Section 3.1.
- DV-2xAA: a method that solves SAV by building on top of the Lazy AA method discussed in Section 3.2.1. In other words, this method first predicts, for both unlabelled documents, who the author of the document is, and then checks if the two predicted authors are the same author.

We consider the following baseline systems:

- STD-CosDist: This consists of a binary classifier trained to predict whether the pair belongs to Same or Different, where a pair of documents is represented by a vector of one feature only. The value of this feature is obtained by calculating the distance between the two documents, each represented by a “standard” vector, and where the distance function is the cosine distance. We have also run experiments using the L1 or L2 distances in place of the cosine distance; we omit to report their results since cosine proved the best-performing one. The training set is transformed into pairs following the same policy as in DV-Bin (see Section 4.2).

[22] As detailed in Section 2.1, in open-set SAV one normally assumes that the authors of the two unlabelled documents are *not necessarily* among the authors represented in the training set; in these experiments, we consider the more difficult setting in which the authors of the two unlabelled documents are *strictly not* among the authors represented in the training set.

The classifier thus learns the distance threshold that best separates the Same pairs from the Different pairs.

- STD-2xAA: This consists of a single-label multiclass classifier that operates on standard vector representations and that, as in DV-2xAA, solves SAV by performing closed-set AA for both documents and then checking if the two predicted authors are the same.

Both baselines are equipped with the same learner as our method, i.e., LR optimised by running the usual optimisation process for hyperparameter C.

Figure 2 reports the experimental results we have obtained, displayed in terms of accuracy (on the y axis) as a function of the number of training documents per author (on the x axis), in datasets `IMDB62`, `PAN2011`, and `Victorian` (each corresponding to a different column), at varying number of authors (each corresponding to a different row). The values for combination (`PAN2011`,25,50) are missing since this combination is not feasible, given that in `PAN2011` there are fewer than 50 authors (25 for the closed-set setting and 25 for the open-set setting) with at least 50 training documents each. Coloured dots each represent an average result across 10 experiments, while the colour band frontiers indicate ± one standard deviation from the mean. Table 2 reports the results for the `arXiv` dataset.

The results clearly indicate that the DV-based variants perform well; of the two methods that achieve SAV by running AA on both documents (i.e., the DV-2xAA and STD-2xAA methods), the DV-based method is always better or much better than the standard vector-based method, and the same happens of the two non-AA-based methods. The top-performing method is unquestionably DV-2xAA, which always outperforms (often by a very large margin) all others, for all numbers m of authors and for all numbers q of training examples per author. As for the reason why DV-2xAA outperforms DV-Bin, we conjecture that this may happen because the Lazy AA method uses only evidence conveyed by few relevant training documents (the k documents most similar to the test document, for both test documents), thus filtering out other less relevant documents; this is in keeping with the fact that methods based on nearest neighbours, as our DV-2xAA method, always pick, during their parameter optimisation phase, values of k that are much smaller than the entire size of the training set.

All algorithms obviously improve their performance as the number of documents per author increases, with the sole exception of STD-CosDist. This latter fact might indicate that the optimal distance threshold that STD-CosDist finds is fairly stable, and is well estimated even by using few training data. However, it seems clear from these results that distances alone do not carry as much information as DVs do.

Figure 3 shows the distribution of $\Pr(\text{Same}|x', x'')$ values for Same and Different pairs that STD-CosDist and DV-Bin compute. For this experiment, we have set $m = 20$ and $q = 50$ for all datasets except for `arXiv`, where we have set $m = 50$ and used all the documents written by the 50 authors. (Note that the "2xAA" variants do not compute a single posterior probability and are thus not amenable to a similar analysis.) The STD-CosDist method manages to separate the posteriors of the Same and Different pairs to some extent in the `IMDB62` and `Victorian` datasets, but it fails to separate them well in `PAN2011` and `arXiv`. Interestingly enough, the posteriors generated by STD-CosDist are close to being normally distributed, both for the Same pairs and for the Different pairs. Things are very different for the DV-Bin method, which tends to generate much more polarised scores (i.e., separate the positives from the negatives much better), placing most of the density mass around 0 for Different pairs and around 1 for Same pairs, which is indicative of a very good performance. Still, the score distribution generated for `Victorian` and, especially, for `PAN2011`, reveal that the DV-Bin method still has room for improvement.

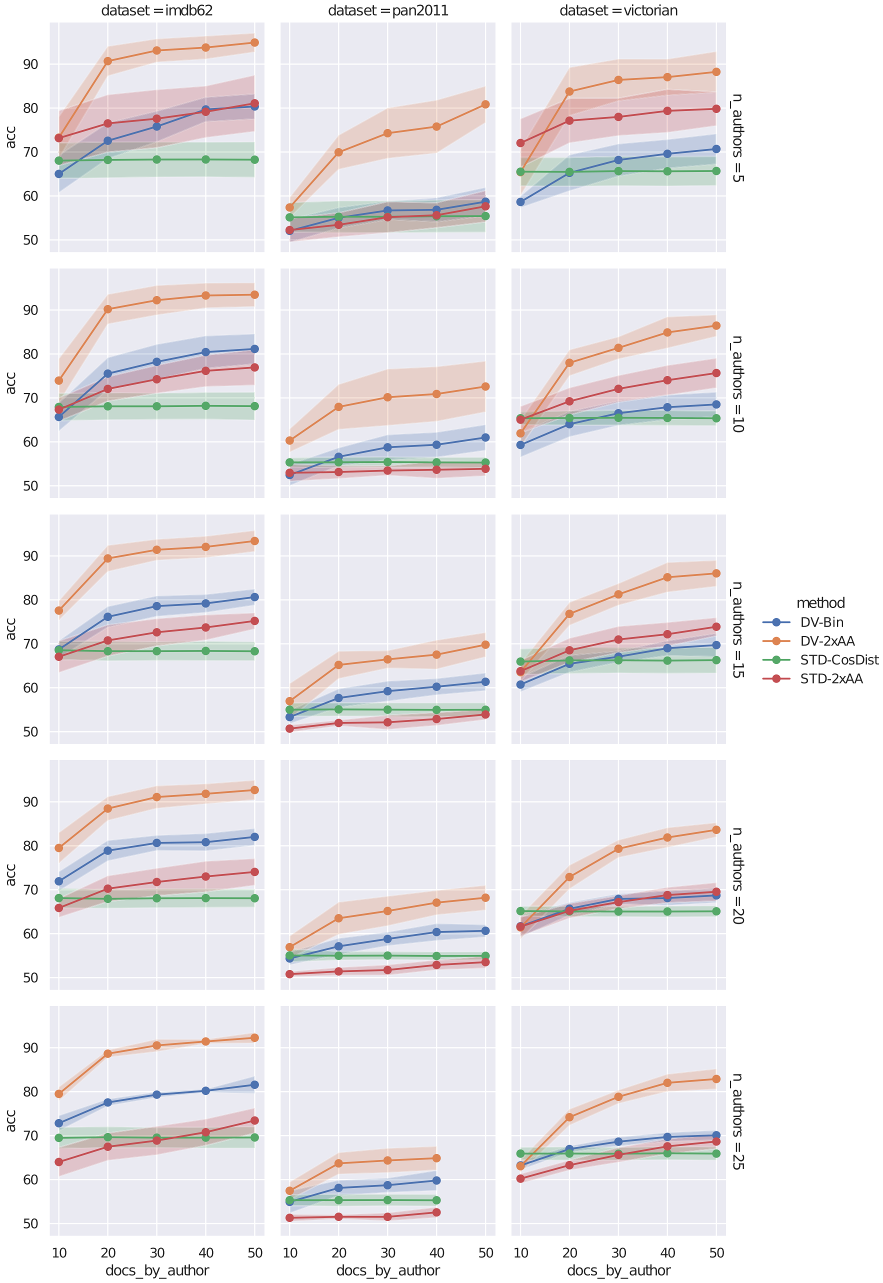

Fig. 2. Intrinsic evaluation of DVs: results on closed-set SAV, using vanilla accuracy (on the y axis) as the evaluation measure on datasets `IMDB62`, `PAN2011`, and `Victorian`.

4.4.2 Experiments on Open-Set SAV. In the open-set SAV experiments, there is no intersection between the set of m authors that we draw to compose the test set and the set of m authors observed during training. This aspect automatically rules out any attempt to perform SAV via authorship attribution (i.e., DV-2xAA); for this reason, in this setting, the only DV-based method we test is DV-Bin. The baseline systems we consider are:

Table 2. Intrinsic Evaluation of DVs: Results on Closed-Set SAV, Using Vanilla Accuracy as the Evaluation Measure on Dataset `arXiv`

	mean	std	ttest
DV-Bin	0.756	0.017	
DV-2xAA	**0.803**	0.025	
STD-CosDist	0.629	0.022	
STD-2xAA	0.646	0.014	

Boldface indicates the best method. Symbols * and ** denote the method (if any) whose score is *not* statistically significantly different from the best one at $\alpha = 0.05$ (*) or at $\alpha = 0.001$ (**) according to a paired sample, two-tailed t-test. No symbols * and ** appear in this particular table since all differences are statistically significant.

- STD-CosDist: This is the same distance-based method that we have used in the closed-set SAV experiments. In this case, the method is constrained to learn the optimal threshold from authors different from those in the test set.
- Impostors: This is a method developed by Koppel and Winter [29]. We use our own implementation of the "blogger's" variant, which had proved superior to others in the experiments of [29] and amounts to using documents from the same domain (blogs in the original authors' experiments, documents from the training set in our case) as the impostors candidates. We use cosine as the distance function since, in our experiments, we have found it to consistently deliver better results than the "minmax" criterion (the similarity function of choice in [29]). We set parameter I (the number of impostor candidates) to 50 instead of 250 (which was found to work well by Koppel and Winter [29]) since our training sets are much smaller than those they considered (sticking to $m = 250$ would basically result in a random choice of impostor candidates); following [29], the rest of the parameter values we use are $i = 10$ (number of impostors) and $k = 100$ (number of bagging trials). Also, following [29], we optimise parameter σ^* (the decision threshold) on a validation set. Note that we have not used this baseline method in the closed-set SAV experiments, since, in that case, the "impostors" cannot be created.

Figure 4 displays the experimental results we have obtained on `IMDB62`, `PAN2011`, and `Victorian`, while Table 3 reports the results obtained for the `arXiv` dataset.

There is no clear winner in the light of these results. DV-Bin seems to perform best in `IMDB62`, especially when the number of authors increases; all methods seem to perform comparably in `PAN2011` and `arXiv`, and STD-CosDist seems to perform slightly better in `Victorian`. Somehow surprisingly, the Impostors method seems not to take advantage of the increase in the number of documents per author, likely because the number of actual impostors ($i = 10$) is set in advance and thus the method is indifferent to variations in q. DV-Bin tends to perform poorly when the number of documents per author is very small (i.e., 10); this may be explained by the fact that the number of Same pairs that can be generated from 10 elements is relatively small. Concerning STD-CosDist, it proves a fairly stable method, as in the closed-set scenario. `PAN2011` proves the hardest dataset here, with all methods performing only marginally better than a random classifier (which would obtain an expected accuracy of 0.50). Regarding the `arXiv` dataset, DV-Bin performs

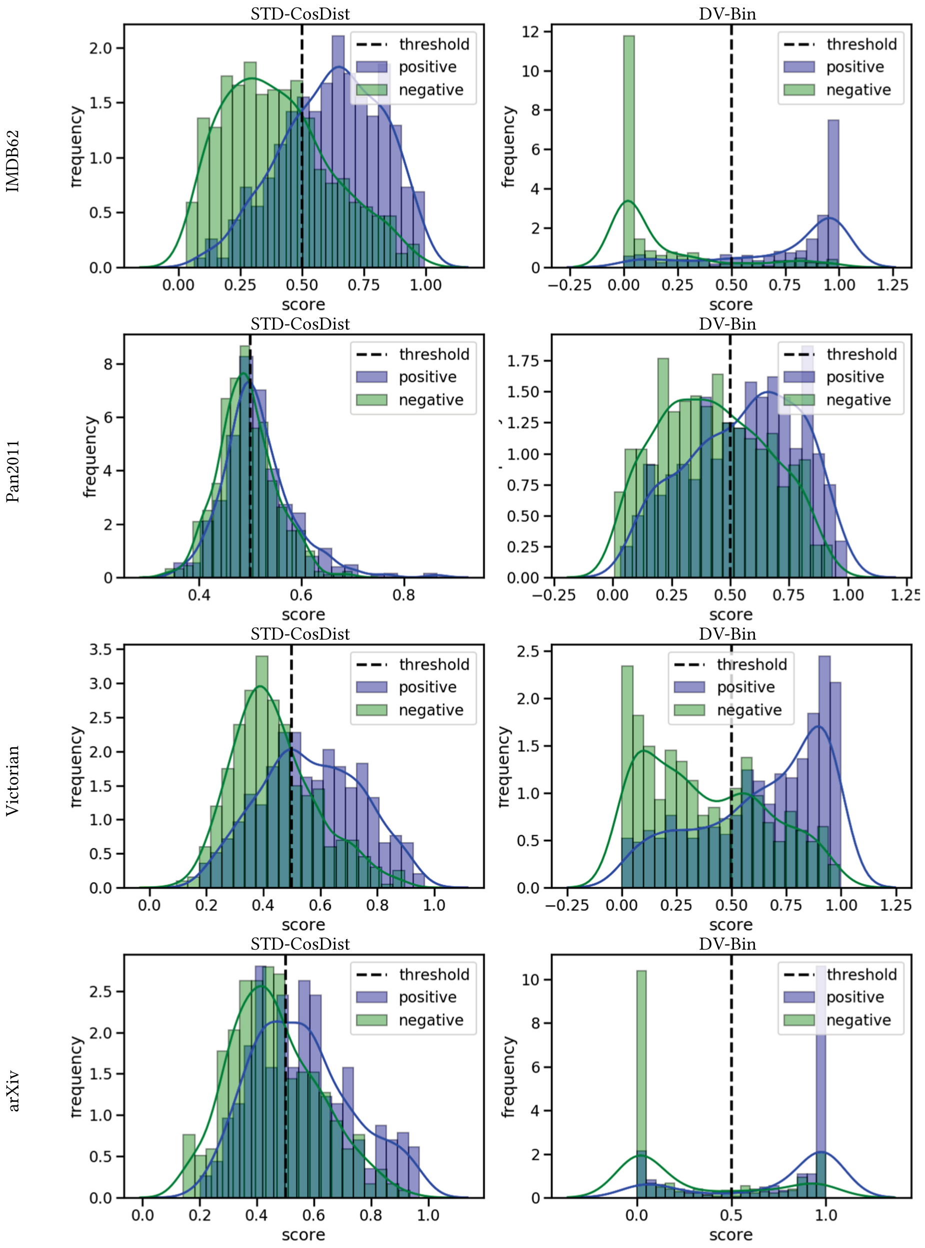

Fig. 3. Distribution of $\Pr(\text{Same}|x', x'')$ values for Same and Different pairs as computed by STD-DistCos (first column) and DV-Bin (second column).

best on average, but the t-test reveals that this superiority is not significant from a statistical point of view.

Summing up, there is no strong enough empirical evidence to claim that the DV-Bin method outperforms Impostors in open-set SAV. However, there are some technical reasons why one should prefer the DV-Bin method to the Impostors method. The first concerns its efficiency. Impostors is

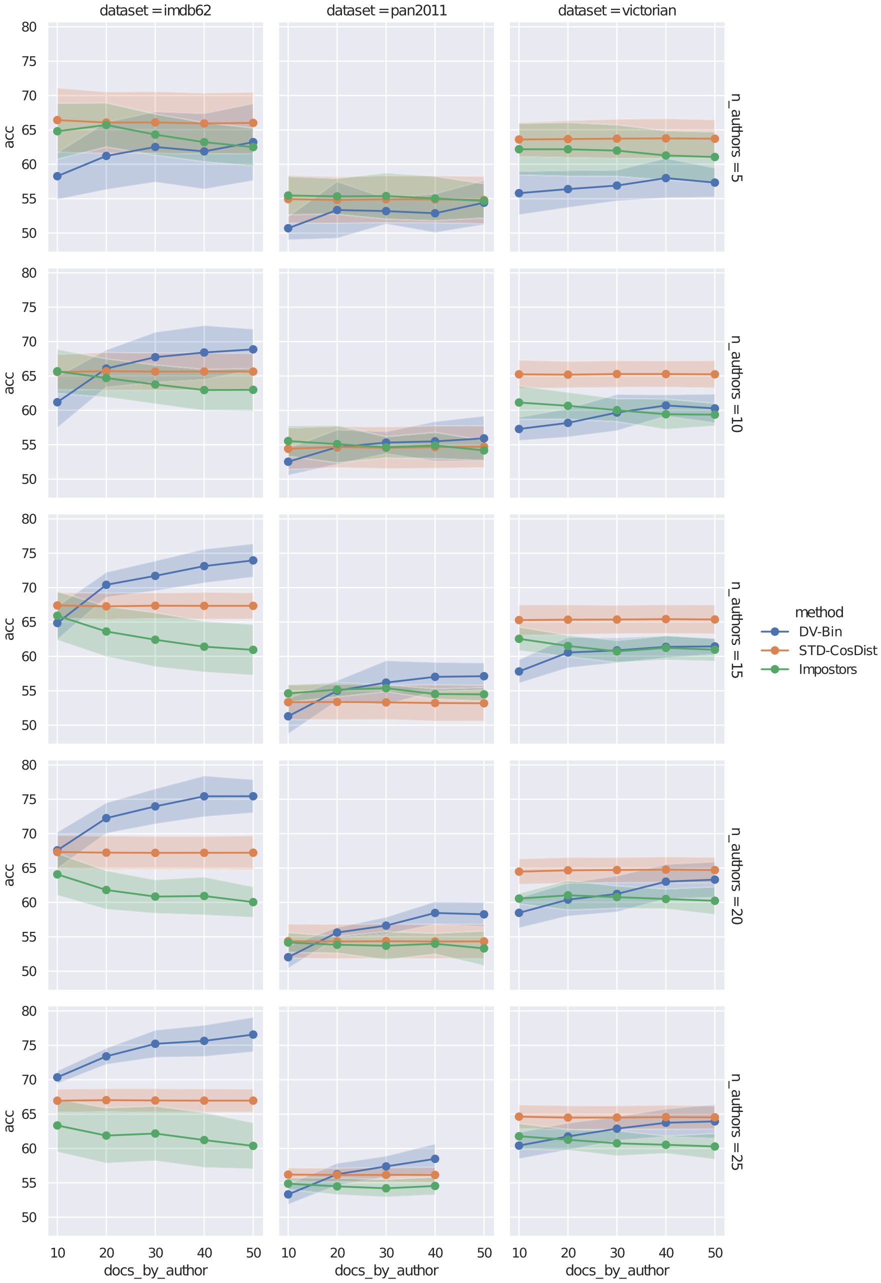

Fig. 4. Intrinsic evaluation of DVs: results on open-set SAV, using vanilla accuracy (on the y axis) as the evaluation measure on datasets IMDB62, PAN2011, and Victorian.

a lazy method, meaning that it has no offline training phase, i.e., all inductive inference is carried out in the classification phase, and the workload that a single test pair entails is significant, since it involves computing the similarity between the test document and each training document, and computing k rounds of bagging for each impostor and for each element in the pair. Conversely,

Table 3. Intrinsic Evaluation of DVs: Results on Open-Set SAV, Using Vanilla Accuracy as the Evaluation Measure on Dataset arXiv

	mean	std	ttest
DV-Bin	**0.663**	0.020	
STD-CosDist	0.661	0.019	**
Impostors	0.642	0.025	**

The notational conventions are the same as in Table 2.

once trained, classifying an unlabelled pair using Diff-Vectors comes down to computing a simple linear combination of feature differences.[23] The second reason concerns its applicability. By definition, the Impostors method cannot be used, as observed above, in closed-set SAV and, more generally, in SAV settings in which documents written by any of the authors of the test pair are observed in training. The reason is that the method would likely consider training documents by one of the test authors as candidate impostors (since these training documents are expected to be more similar to the test document), and thus the test author could wrongly be taken for an impostor of herself.

Figure 5 shows the distribution of the decision scores (i.e., the posteriors $\Pr(\text{Same}|x', x'')$) for Same pairs and Different pairs that Impostors and Diff-Vectors compute. As for closed-set SAV, we set $m = 20$ and $q = 50$ for all datasets except arXiv, for which we instead set $m = 50$ and keep all documents per author. Recall that, in our open-set setting, m specifies both the number of authors involved in the training set *and* the number of authors involved in test (e.g., in the case of arXiv, we are using the entire dataset since there are 100 distinct authors). For ease of visualisation, we report the score values according to a logarithmic scale.

The Impostors method produces decision scores which tend to be very close to 0. The dashed vertical line indicates the decision threshold found optimal in the validation phase; this threshold is $\sigma^* = 0.005$ in all cases but in arXiv, where $\sigma^* = 0.01$ worked better. Note that this threshold succeeds in placing most of the negative scores below it, but still misclassifies many positives. Particularly, in PAN2011 and arXiv, it fails to push many of the positive scores beyond the decision threshold.

The DV-Bin method instead succeeds at polarising the decision scores of Same and Different pairs in IMDB62 and arXiv, although it fails to allocate most of the negative mass below the 0.5 threshold in Victorian and, to a greater extent, in PAN2011.

Overall, as clear from a simple visual inspection, the DV-Bin method is better than the Impostors method at correctly separating the scores of the Same pairs from those of the Different pairs on each of our four datasets.

4.5 Extrinsic Evaluation of Diff-Vectors

Our "extrinsic" evaluation of DVs consists of closed-set AA experiments. We do not run experiments for AV since, as discussed in Section 3.3, each of our AA experiments is also a set of m AV experiments, and can be evaluated as such.

[23] Of course, it is fair to mention that the Impostors method incurs no cost for training. But this only applies if the value of the parameter σ^* is hard-wired. In practice, the optimal σ^* has to be estimated in a validation phase, which amounts to using a training set to perform repeated rounds of tests which, as indicated above, require a considerable computational effort.

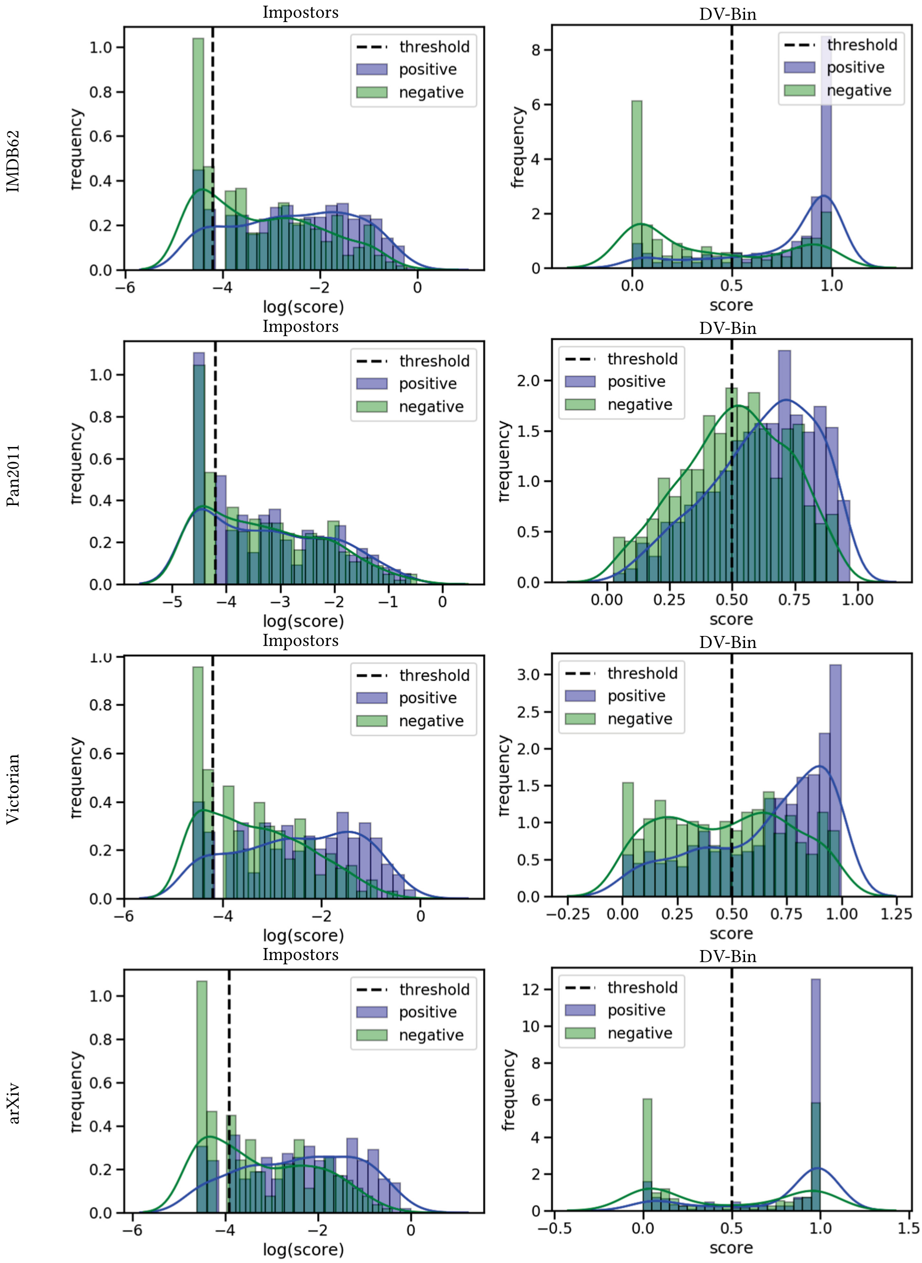

Fig. 5. Distribution of decision scores for positive and negative (i.e., Same and Different) pairs as computed by the Impostors method (1st column; note the log scale) and by the DV-Bin method (2nd column).

4.5.1 The AA Results. At the core of our AA methods is a SAV classifier that operates on pairs of documents. Given a test document x, attribution for it is performed by applying a combination rule to the posterior probabilities generated for pairs of documents consisting of the test document x and a training document x'. In particular, we explore:

- Lazy AA: the lazy combination rule inspired by k-NN discussed in Section 3.2.1.
- Stacked AA: the linear combination rule inspired by stacked generalisation discussed in Section 3.2.2.

In these experiments, we consider a set of authors $\mathcal{A} = \{A_1, \ldots, A_m\}$, with $m > 2$, each one having q training documents. Given a test set $\mathcal{U}$, the method is asked to attribute each test document to one of the authors in $\mathcal{A}$, in a single-label multiclass fashion.

We investigate the impact on AA accuracy of the number m of authors and the number q of documents per author. We let m take values in the set $\{5, 10, 15, 20, 25\}$ as before, and we let q take values in the set $\{5, 10, \ldots, 45, 50\}$.[24] As in Section 4.4, and for analogous reasons, the experiments are different for the arXiv dataset, in which the above fine-grained exploration is not possible. For both Lazy AA and Stacked AA, we use DV-Bin as the underlying SAV mechanism.

In this case, instead of vanilla accuracy, we use F_1 as the evaluation measure, since not all our datasets are balanced, and since vanilla accuracy is, different from F_1, a notoriously bad measure for working with imbalanced datasets. For all datasets, we report the values of macro-averaged F_1, i.e., F_1 averaged across the authors in $\mathcal{A}$; in the case of the arXiv dataset, we also report micro-averaged F_1 (i.e., F_1 as obtained on a global contingency table generated by all the classification predictions for all authors), since this is the only imbalanced dataset of the lot (and since micro-averages would coincide with macro-averages in the perfectly balanced datasets IMDB62, PAN2011, and Victorian). All results are reported as averages across 10 runs that use different random seeds.

As our baseline, we consider STD-AA, a single-label multiclass classifier trained to distinguish among the m classes from the observation of "standard" vectors of features. Given a test document, the classifier returns the author, which obtains the maximum posterior probability.

Figure 6 displays the experimental results we have obtained for the IMDB62, PAN2011, and Victorian datasets (for the moment being let us disregard the curves for STD-Bin, on which we will comment later), while the first three rows of Table 4 report the results obtained on the arXiv dataset.

These results show the drastic superiority of DVs over standard vectors for AA. Only for $q = 5$, and infrequently for $q = 10$, does STD achieve (marginally) better results than the DVs-based variants; in this case, the reason might have to do with the fact that low values of q result in fewer Same pairs (e.g., for $q = 5$ there are only 10 unordered pairs), which might lead to suboptimal accuracy for the underlying SAV methods. Regarding our variants, the k-NN -inspired combination rule consistently outperforms the linear one in IMDB62 and arXiv, and is slightly better or comparable in the rest of the cases. All methods understandably benefit from the increase in q, but DVs seem to do so at a much greater rate; indeed, the increase in the number of training examples is quadratic in q for the DVs-based variants, while it is linear in q for STD.

4.5.2 *The AV Results.*

Concerning the AV task, note that macro-averaged F_1 is also the right measure for evaluating AV; in fact, F_1 as measured on a specific author A^* is the right measure for evaluating AV once A^* is considered the candidate author, and macro-averaged F_1 is the right measure for computing the average performance for all possible choices of A^*. As a consequence, the results reported in Figure 6 and Table 4 also count as an evaluation of the reported methods for the AV task.

[24] The reason why we explore a finer-grain grid for q with respect to our previously discussed SAV experiments is that, in this case, we are not considering Impostors as a competitor, and thus these experiments are considerably faster to run. Note also that, differently from our SAV experiments, we here report experiments also for the combination (PAN2011,25,50) since here we are only considering the closed-set setting, and since in PAN2011 there are at least 25 authors with 50 documents.

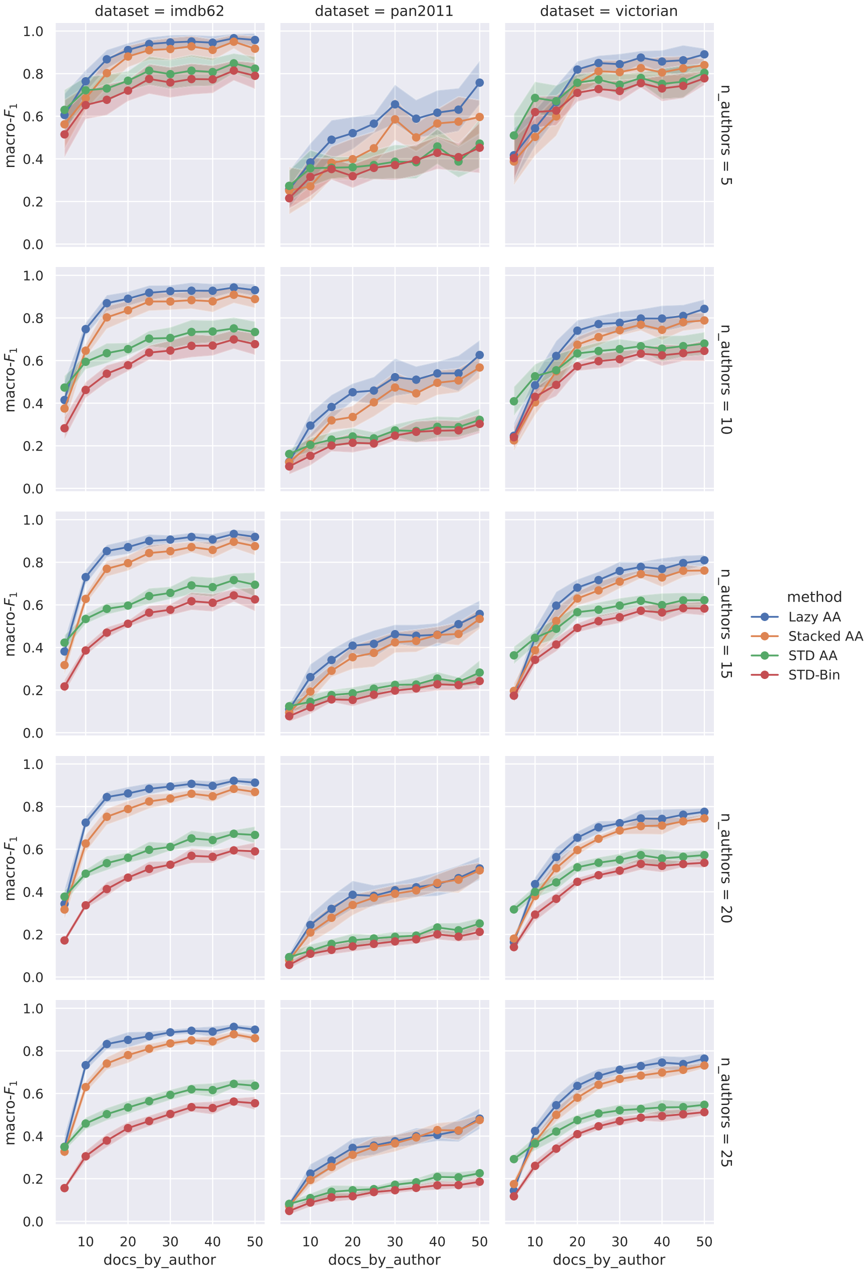

Fig. 6. Extrinsic evaluation of DVs: results on closed-set AA in terms of F_1 for the IMDB62, PAN2011, and Victorian datasets.

For AV, we add another baseline (which we call STD-Bin), which consists of a binary classifier trained to distinguish between A^* and $\overline{A^*}$ from the observation of "standard" vectors of features; it is fair to add this baseline since it would be just natural to solve AV by means of a binary classifier, instead of by means of a multiclass classifier as STD-AA does.

Table 4. Extrinsic Evaluation of DVs: Results on Closed-Set AA in Terms of F_1 for the arXiv Dataset

	macro-F_1			micro-F_1		
Method	mean	std	ttest	mean	std	ttest
Lazy AA	**0.682**	0.024		**0.676**	0.019	
Stacked AA	0.639	0.028		0.639	0.024	
STD-AA	0.374	0.018		0.405	0.021	
STD-Bin	0.235	0.014		—	—	

The notational conventions are the same as for Table 2.

However, the experimental results show STD-Bin to be inferior to STD-AA, as clear from both Figure 6 and Table 4. This is in keeping with the results of our preliminary experiments (discussed in Section 3.3) that had convinced us to abandon the idea of performing AV via Lazy AV and Stacked AV, in favour of versions of Lazy AA and Stacked AA in which we attribute document x to A^* if the AA algorithm does so and we attribute x to $\overline{A^*}$ if the AA algorithm attributes it to an author A_z different from A. Concerning the likely reasons why this happens, the same considerations we made in Section 3.3 apply.

In sum, given that STD-Bin is not a serious contender, the same considerations on the superiority of DV-based methods over standard methods that we had made in Section 4.5.1 for AA also apply to AV.

4.6 Efficiency

The improvements in performance obtained by DV-based methods with respect to methods based on standard vectorial representations can be attributed to the increase in the number of training examples resulting from pairing documents. However, this can be expected to come at a computational cost. In this section, we compare the actual cost of DV-based methods with that of methods based on standard representations.

4.6.1 Efficiency Analysis. Let $n = |\mathcal{L}|$ be the number of training documents. Let us also assume that the total cost of training a classifier is bounded by some function f on the number n of training documents, a cost which depends on the learning algorithm and its implementation; in other words, this total cost is $O(f(n))$, where we can safely assume $f(n)$ to grow faster than or equal to n. (We take the number of features as constant, which means that this number does not impact our analysis of efficiency.) Let us also assume that the classification of a document requires constant time, i.e., is $O(1)$.

The cost of training the DV-Bin classifier of Section 3.1 comes down to the cost of generating the $n(n-1)/2$ pairs, which is $O(n^2)$, plus the cost of training a classifier using $n(n-1)/2$ DVs, which is $O(f(n^2))$. In practice, and in order to keep the computational burden under reasonable bounds, we only generate a fixed number of examples (i.e., we avoid generating all pairs first and discarding some of them later). Let $n = mq$, with m the number of authors and q the number of documents per author, as before; we generate all $mq(q-1)/2$ pairs of type Same and as many pairs of type Different, thus ending up with $mq(q-1)$ documents, which has a cost $O(mq^2) = O(nq)$, plus, again, the cost of training the classifier from the $mq(q-1)$ documents, which is $O(f(nq))$; since we have assumed $f(n)$ to grow faster than or equal to n, the total cost of generating a DV-Bin classifier is $O(f(nq))$. At classification time, we only need to compute the absolute difference between two vectors and invoke the classifier; for most classifiers (and for LR in particular) this cost can be considered constant, i.e., $O(1)$.

Table 5. Computational Cost of a Number of Algorithms Discussed in this Article

	Tasks	Training	Test
STD	AV, AA	$O(f(n))$	$O(1)$
DVs	SAV	$O(f(nq))$	$O(1)$
Lazy AA	AV, AA	$O(n^2 \log q)$	$O(n \log q)$
Stacked AA	AV, AA	$\max\{O(n^2), O(f(nq))\}$	$O(n)$
Impostors	SAV	$O(n \log n)$	$O(n \log n)$

As a lazy algorithm, Lazy AA does not involve any real training phase. However, it seeks for the optimal value of k, and this entails pre-computing a matrix of distances, which is done only once (and is $O(n^2)$), plus sorting, for each of the m authors and for each of the n training documents (see Equation (8)), the q training documents by this author (which is $O(q \log q)$). Altogether, this entails a total cost of $O(n^2 + mnq \log q) = O(n^2 \log q)$. At classification time (for both the AA and the AV settings), we only need to sort, for each of the m training authors, the q training documents by this author, which means that this is $O(mq \log q) = O(n \log q)$.

Concerning Stacked AA, training the system entails (i) training a DV-Bin classifier, which, as argued above, has a cost $O(f(nq))$; (ii) creating the projections $\phi(x)$ for each of the n training documents, which has a cost $O(n^2)$ (since creating one such projection has a cost $O(n)$); (iii) training the metaclassifier on the n vectors $\phi(x)$ thus generated, which has a cost $O(f(n))$; the total cost of training the system is thus the larger of $O(n^2)$ and $O(f(nq))$. At classification time, we need to generate the representation $\phi(x)$ of the test document, which has a cost $O(n)$, and to invoke the meta-classifier, which we can assume to require constant time.

The Impostors method does not properly carry out a training phase, but incurs the cost of optimising the σ parameter, which consists of carrying out t rounds of test, with t a user-defined parameter. The computational cost of testing whether two documents have been written by the same author or not entails computing, for each of the n training instances, the similarity with each test document, which is $O(n)$, plus sorting by similarity in order to choose the "impostors", which is $O(n \log n)$; this means that the total cost is $O(n \log n)$. Impostors then perform k rounds of bagging trials with respect to each of the i impostors, which adds a cost $O(ki)$ if we assume the similarity function to be computed in constant time.

Table 5 summarizes all the costs involved.

4.6.2 Timings. As for the experiments reported in Figures 3 and 5, we report actual timings clocked for $m = 20$ and $q = 50$ in the case of the IMDB62, PAN2011, and Victorian datasets, and for the entire dataset in the case of arXiv. The variables that influence the analysis include the number of training documents ($|\mathcal{L}|$), the number of pairs generated by DVs ($|\mathcal{L}_{\mathcal{P}}|$), and the number of test documents ($|\mathcal{U}|$). Recall that $|\mathcal{L}_{\mathcal{P}}|$ depends on the number of Same pairs that can be generated, which is fixed and amounts to $20(50 \cdot 49)/2 = 24{,}500$ for IMDB62, PAN2011, and Victorian, and which is variable and depends on the random split (we report the value averaged across 10 runs) for arXiv. The values are summarised in Table 6 for convenience. Recall that the number of test pairs in SAV tasks is fixed for all datasets and is equal to 1,000. Note that the arXiv dataset is split differently for SAV and AA since, although we used the entire dataset in both tasks, in the former, we held half the authors out for composing the open set. All times refer to computations carried out on the same machine, equipped with a 12-core processor Intel Core i7-4930K at 3.40 GHz with 32 GB of RAM, under Ubuntu 18.04. All methods run on CPU and are implemented using

Table 6. Size of the Datasets Used for the Efficiency Test

	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{L}_{\mathcal{P}}\rvert$	$\lvert\mathcal{U}\rvert$
`IMDB62`	1,000	49,000	6,000
`PAN2011`	1,000	49,000	463
`Victorian`	1,000	49,000	7,937
`arXiv-SAV`	518	5,784	255
`arXiv-AA`	1,028	11,106	441

Table 7. Training and Testing Times (in Seconds) Clocked When Solving the SAV Task

	`IMDB62`		`PAN2011`		`Victorian`		`arXiv-SAV`	
	Train	Test	Train	Test	Train	Test	Train	Test
DV-Bin	438.3	1.4	173.7	**0.7**	870.2	2.5	49.8	0.3
STD-CosDist	**6.7**	**0.3**	**3.5**	0.8	**11.9**	**0.3**	**0.7**	**0.2**
Impostors	271.9	625.1	247.6	455.7	283.2	651.3	225.2	224.6

Boldface and underlining indicate the fastest and slowest methods for each dataset, respectively.

`scikit-learn` and the `SciPy` stack. We have parallelised all parallelisable steps, both in training and test, for all algorithms.

Table 7 reports the average time each method requires to complete the SAV task, both in terms of training time and testing time for each dataset. The method that uses standard vectors to compute the cosine distance (STD-CosDist) is much faster than any competing method, both in terms of training times and test times. This is due to the fact that cosine can be computed very quickly, and that the classifier operates on one single feature. At training time, both DV-Bin and Impostors are computationally much more expensive, with neither one being clearly better than the other. However, at classification time DV-Bin is much faster than Impostors, and costs no more than a few seconds to accomplish the 1,000 SAV computations, comparably to STD-CosDist. Impostors, on the contrary, require much more time, and its testing times are higher than its training times. (Recall that, for the Impostors method, by “training” we mean the search for the optimal value of parameter σ by using the training set, since Impostors does not properly perform any training.)

Table 8 reports the average time each method requires to complete the AA Task. It is immediately evident that, at least on `IMDB62`, `PAN2011` and `Victorian`, the two most expensive methods are the ones based on DVs, both at training time and at classification time (neither one is systematically better or worse than the other, though); the STD method is thus almost always the fastest. The reason for this high computational cost of the DV-based methods is that, despite the fact that DV-Bin proved very fast at classification time in SAV, Lazy AA, and Stacked AA invoke DV-Bin *many* times, i.e., require computing, for all training documents (in the training phase) and for all test documents (in the testing phase), the similarity (viewed as a posterior probability computed by DV-Bin) with each training document. This has an important impact both in the training phase and in the testing phase.

Although the increase in training time with respect to the SAV task is not marked, the penalty paid during the testing phase is instead evident; for `arXiv`, training times of the DV-based methods increase substantially with respect to those seen for the SAV task, which is due to the fact that, in this case, the training set is twice as large as that for SAV—see Table 6). In some cases (`IMDB62` and `Victorian`) testing times even surpass training times; this can be explained by the fact that

Table 8. Training and Testing Times (in Seconds) Clocked for Solving the AA Task

	IMDB62		PAN2011		Victorian		arXiv-AA	
	Train	Test	Train	Test	Train	Test	Train	Test
Lazy AA	460.2	955.5	228.3	61.8	917.4	1232.6	**145.0**	66.1
Stacked AA	480.3	942.8	267.3	61.2	955.4	1235.9	250.4	66.5
STD-AA	**156.0**	**0.2**	**157.6**	**0.1**	**202.9**	**0.5**	483.8	**0.1**

Boldface and underlining indicate the fastest and slowest methods for each dataset, respectively.

those datasets contain the largest test sets (6,000 and 7,937 instances, respectively), which means that computing the matrix of posterior probabilities becomes especially costly.

Somehow surprisingly, though, the variants based on DV-Bin were trained faster than STD in arXiv; the reason for this lies in the number of authors involved, which in this dataset is the largest, i.e., $m = 100$. STD thus needs to train 100 binary classifiers on a document-by-feature matrix of $O(10^5)$ dimensions, while DV-based variants need to train only one binary classifier in order to discern between Same or Different; note also that, in this case, the number of pairs generated is comparatively smaller than for other datasets. The rest of the work that DV-based methods undertake is on a matrix of posterior probabilities that has just $|\mathcal{L}|$ = 1,028 dimensions in the case of arXiv; training 100 binary classifiers in Stacked AA is thus much faster than with STD. To conclude, DVs bring about substantially higher computational costs than the "standard" representations, both at training time and at test time.

4.6.3 Are the Costs Incurred by DVs Tolerable? The conclusions reached in the previous section bring up the issue whether we should look for simplifications of our algorithms that cut down on training and/or classification time, maybe at the cost of some reduction in accuracy.

One example is the use of Stacked AA for performing AV. In the current setting, in order to perform AV (i.e., to choose, for a disputed document $\mathbf{x}$, between a candidate author A_i and its complement $\overline{A_i}$), (i) we invoke Stacked AA in order to decide who among the authors of $\mathcal{A} = \{A_1, \ldots, A_i, \ldots, A_m\}$ is the true author of $\mathbf{x}$, and (ii) we assign $\mathbf{x}$ to A_i if Stacked AA predicts A_i to be the author of $\mathbf{x}$ and to $\overline{A_i}$ if Stacked AA predicts an author in $\{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m\}$ to be the author of $\mathbf{x}$. In other words, in order to solve the (binary) AV problem, we first invoke a multiclass algorithm (Stacked AA) and then convert its multiclass decision into a binary decision. A much more efficient way would be to use Stacked AV (see Section 3.3 for details), which directly solves a binary problem by means of a binary algorithm. However, as discussed in Section 3.3, preliminary experiments that we had run had shown this "direct" method to be less accurate than the "indirect" method we adopted. Other simplifications of Lazy AA or Stacked AA that we have come up with are, as in the case above, more efficient but less accurate.

As often in classification, there is thus a tradeoff between efficiency and accuracy. We have decided to stick to our methods, in the forms described in Sections 3.2.1 and 3.2.2, and to avoid the simplifications discussed above, because we think that, in authorship identification, accuracy should not be compromised on the altar of efficiency. There are three reasons for this.

The first reason is that, in real authorship identification cases, increased classification times are usually tolerable, because typical such cases *do not involve many unlabelled documents*. Indeed, there is often a *single* unlabelled document, of extremely high value, that we need to make a prediction for (e.g., a text of literary value [5, 9, 35, 41, 46], or an anonymous letter), and in this case issuing a prediction in milliseconds or in minutes does not make a big difference.

The second is that the decisions of an authorship identifier are often of paramount importance (typical examples are its applications in cultural heritage, in cybersecurity, and in digital

Table 9. Results, in Terms of Macro-F_1, Obtained by Applying AA Methods to “Native” AV Problems

	IMDb62			PAN2011			Victorian			arXiv		
	mean	std	ttest	mean	std	ttest	mean	std	ttest	mean	std	ttest
Lazy AA	0.287	0.044		0.166	0.019		0.205	0.010		0.234	0.037	
Stacked AA	0.664	0.062	**	0.278	0.056	**	0.619	0.044	**	0.365	0.075	**
STD-Bin	**0.683**	0.045		**0.285**	0.028		**0.639**	0.042		**0.432**	0.097	

forensics), so no user would prefer to increase the risk of a misidentification for the sake of efficiency.

The third reason is that classifier training is performed once for all, and the times reported in Tables 7 and 8 are plausible for most application contexts. Note also that, in most authorship analysis applications, training documents are scarce, which means that scenarios in which the training documents are many more than in our datasets (which would mean training times higher than those reported in Tables 7 and 8) are unfortunately infrequent.

All this indicates that reduced (training and/or classification) efficiency is not a critical issue in authorship identification, and that this reduced efficiency is tolerable if it leads to higher accuracy.

4.7 Can We Use Diff-Vectors for “Natively Binary” AV Problems?

So far, we have tested DVs in situations in which, at training time, we assume we know who among the n authors in $\mathcal{A}$ has written which training documents. That is, we have recast SAV, AA, and AV in terms of a multiclass task. In this section, we turn to analyse experimentally the suitability of DVs for AV in a different situation, i.e., one in which all we know about a certain training document is whether it has been written by the author of interest or not, that is, whether this document is a positive example or a negative example with respect to a binary classification scheme.

To this aim, we randomly draw $m = 10$ authors for each dataset, and perform, for each author, an AV experiment in which we take this author as the positive class and the rest of the authors (grouped together) as the negative class, and where we employ an AA method (i.e., one that was originally devised for tackling arbitrary values of n) for the particular case of $n = 2$. That is, given $\mathcal{A} = \{A_1, A_2, \ldots, A_{10}\}$, we generate a binary setting $\mathcal{A}'_i = \{A_i, \overline{A}_i\}$, and we do this for all authors by letting i vary in the range $\{1, \ldots, 10\}$. In each of these experiments, we take $q = 50$ documents for each author in $\mathcal{A} = \{A_1, A_2, \ldots, A_{10}\}$ in all datasets but in arXiv, for which we take all the documents available for the author. We repeat the entire process 10 times with different random seeds and report results averaged across all experiments. The results reported in Table 9 show that, in such a setting, DVs do not bring about any benefit.

This was somehow to be expected, since DVs bring useful additional evidence to the learning process for AV only when we have access to the entire labelling information, i.e., when author A_j can help improve classification for author A_i indirectly, by strengthening the internal SAV function with additional instances of the class Same that come from documents written by A_j (see Section 2.3). This is not possible in a pure binary setting, since the negative class *is not homogeneous* (i.e., it does not represent the production of one single author, but the production of many authors that have been mixed together), and thus cannot be leveraged to generate positive instances for the class Same. For similar reasons, we cannot generate instances of Different by simply picking two documents from the negative class, since those could have been written by the same (unknown) author. What we are left with, thus, is the possibility to generate $q(q-1)/2$ instances of Same only from pairs of instances from the positive class A_i, and $q^2(m-1)$ instances of Different by generating pairs in which one document has been written by A_i and the other by $\overline{A}_i$. Since the positive evidence for the surrogate SAV problem (class Same) comes exclusively from the positive class of the AV problem (author A_i), there is no real information gain with respect to using standard representations. Indeed, we observe a degradation in performance of both variants with respect to the adoption of "standard" vector representations; this degradation is not statistically significant for the stacking variant, though.

For this reason, we conclude that, in AV settings, DV-based methods should be used only in situations in which we have access to the entire class label information. Luckily enough, access to the entire class label information is something that characterises most scenarios in which AV is to be applied since, when investigating whether document x is indeed by author A^* or not, it makes sense to generate a training dataset in which negative instances are known to be by authors "close" (in a stylistic sense) to A^*, and we can know this only by knowing who the author of each document is.

4.8 Results with a Different Learner

In this section, we discuss some additional experiments that we have run in order to check whether the DV-based methods are superior to methods based on standard vectors also when using a learning algorithm different from LR. For these experiments, in which we compare the Stacked AA DV-based method with the "standard" STD-AA method (see Section 4.5.1), we have thus switched to support vector machines (SVMs), since they are frequently used in the authorship analysis field (see e.g., [10, 11, 15, 24, 47, 53]), where they have usually shown very good performance. Similarly to LR, we use the implementation of (linear) SVMs available from the scikit-learn library;[25] as customary in the text learning field, we use a linear kernel. In order to speed up the computation, we simply rely on default hyper-parameters and perform dimensionality reduction using "lightweight random indexing" [32], projecting the original space onto a 1,000-dimensional space. This attempt at reducing the computational cost of the process is justified by the fact that SVMs are not probabilistic classifiers, and their outputs need thus be converted into posterior probabilities; this conversion is attained by means of "calibration", a process that incurs the additional cost of performing model selection via cross-validation. For this, we set the number of folds to 3 and adopt Platt scaling as our calibration method.

The macro-F_1 and micro-F_1 results reported in Table 10 are obtained by running the corresponding setup for $m = 25$ randomly chosen authors, for which we randomly choose $q = 50$ training documents per author (except for the arXiv dataset, for which we use all data available). We report the mean results, along with their standard deviation, across 10 runs performed with different random seeds.

As evident from Table 10, these results clearly confirm the conclusions that we had drawn from the LR experiments (see Section 4.5.1), and indicate that the Stacked AA DV-based method is largely superior to the "standard" STD-AA method; indeed, the former outperforms the latter on all four datasets and for both evaluation measures, always by a large margin.

As a final note, we recall from Section 4.5.2 that the macro-F_1 results of an AA experiment also count as an evaluation of the tested methods for the AV task, which indicates that, also when using SVMs as the learning algorithm, DV-based methods are superior not only in terms of AA but also in terms of AV.

5 RELATED WORK

In the authorship analysis literature, some example works (starting from Koppel and Winter [29]) in which two or more documents are represented by a single vector have been presented before; the main difference between those papers and the present one is that none among the former

[25] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Table 10. Macro-F_1 and Micro-F_1 Results Obtained on the Authorship Attribution Task with a Different Supervised Learning Method (SVMs); the LR Results are Included for Comparison Purposes, and are Repeated from Section 4.5.1

Measure	Learner	Method	IMDB62	PAN2011	Victorian	arXiv-AA
Macro-F_1	LR	Stacked AA	**0.8596** ± 0.0163	**0.4754** ± 0.0256	**0.7315** ± 0.0131	**0.6386** ± 0.0277
		STD-AA	0.6365 ± 0.0295	0.2259 ± 0.0161	0.5471 ± 0.0172	0.3741 ± 0.0178
	SVMs+LRI	Stacked AA	**0.8270** ± 0.0226	**0.4313** ± 0.0323	**0.6549** ± 0.0276	**0.5655** ± 0.0227
		STD-AA	0.6213 ± 0.0335	0.2095 ± 0.0263	0.4722 ± 0.1236	0.3177 ± 0.0220
Micro-F_1	LR	Stacked AA	**0.8599** ± 0.0158	**0.5546** ± 0.0331	**0.7509** ± 0.0163	**0.6395** ± 0.0236
		STD-AA	0.6406 ± 0.0286	0.2682 ± 0.0231	0.5795 ± 0.0276	0.4045 ± 0.0213
	SVMs+LRI	Stacked AA	**0.8233** ± 0.0217	**0.5066** ± 0.0376	**0.6928** ± 0.0305	**0.5889** ± 0.0184
		STD-AA	0.6381 ± 0.0307	0.2660 ± 0.0313	0.5183 ± 0.1230	0.3857 ± 0.0198

performed any systematic study, as we instead do, of the implications of the use of these representations. In the next paragraphs, we summarise the major approaches along this line.

As previously mentioned, [29] was the first work in which vectors each representing more than one document were used in the authorship analysis literature. In this representation, a vector represented two documents, the label of the vector was either Same or Different, character 4-grams were used as features, and the value of each feature was the absolute difference between the tf-idf weights of the feature in the two documents. As mentioned in the introduction, the goal of Koppel and Winter [29] was to propose a different method (the "impostors" method for SAV), and they dismiss the DV-based representation as a "simplistic baseline method" [29, p. 179].

Since then, a number of authors started to view the AV task in terms of predicting whether vector $f(X_{A^*}, x)$ belongs to class Same or to class Different, where $f(X_{A^*}, x)$ is a vector derived from the entire set (here represented as X_{A^*}) of training documents known to be by candidate author A^*, and from the document of unknown paternity (here noted as x). For instance, in Reference [4], vector $f(X_{A^*}, x)$ is a vector in which each feature value is the absolute difference between the value of the feature in x and the mean of the values of the feature across the documents in X_A. A slightly different approach is used in the PRNN method presented by Hosseinia and Mukherjee [18]. They view X_{A^*} as a document (generated by the concatenation of all the documents in it), use both this document and document x as input for a parallel neural network composed of an embedding layer and an RNN layer, and combine the two outputs by computing a vector $f(X_{A^*}, x)$ consisting of values of similarity between the two documents. The same work also proposes a different method, called TE, which is based on a transformation encoder that transforms the vector representing A^* into the vector representing x, and takes the resulting loss as a measure of similarity; the authors repeat the process several times using different feature sets, and generate a vector $f(X_{A^*}, x)$ consisting of the different similarity values. In a similar vein, in Reference [6], the $f(X_{A^*}, x)$ vector is composed of 7 similarity values computed on the char n-grams of the two documents. Unlike the present work, none of the above works attempts to tackle the AV and AA tasks by recasting them in terms of SAV.

More recently, [19, 31, 48] tested the use of DVs for the open-set SAV problem at the PAN2021 shared task; in particular, Menta and Garcia-Serrano [31] propose a method that feeds DVs to a double-channel neural network, where the feature values are the tf-idf weights of character n-grams in one channel, and of punctuation marks in the other channel. The outputs of the two channels are then concatenated in a final series of layers, that ultimately leads to the classification decision.

Finally, we note that the DV-based representations that we have discussed are reminiscent of ideas that have been independently explored in multilingual text classification. In particular, Moreo et al. [32] investigate the idea of applying lightweight random projections to the feature space.

Mathematically, a random projection XR of a matrix $X \in \mathbb{R}^{np}$, with n the number of documents and p the number of features, can be attained by multiplying it with a random matrix $R \in \mathbb{R}^{pr}$, with $r \ll p$ the number of dimensions. The term "lightweight" refers to the fact that the rows in R contain only two non-zero values (−1,+1). The pair-based version $\mathcal{L}_{\mathcal{P}}$ of a dataset $\mathcal{L}$ can be defined in terms of $|R \cdot X|$, where $X \in \mathbb{R}^{np}$ is our document-by-feature matrix and R is instead a lightweight projection matrix R^{rn}, this time with r, the number of pairs, much higher than n; here $| \cdot |$ represents the element-wise absolute value. Such a projection effectively computes the absolute difference between two chosen documents.

6 CONCLUSION

In this work, we have discussed the implications of the use of Diff-Vectors (DVs) in authorship identification tasks. A DV is a vector that represents a pair of documents in such a way that the value of a feature in the DV is the absolute difference between the relative frequencies (or increasing functions thereof) of the feature in the two documents. DVs were originally introduced by Koppel and Winter [29], but in that work, the authors dismissed DVs as a "simplistic baseline method"; neither Koppel and Winter [29] nor other authors studied the implications of the use of DVs in authorship identification. A systematic study of these implications is what this article describes.

DVs are naturally geared towards solving the "SAV" task, i.e., the binary task of deciding whether two documents have been written by the Same (possibly unknown) author or by Different authors. However, we have shown that both (i) (closed-set) AA (the task of predicting who among a given set of candidates is the true author of a given text), and (ii) authorship verification (the task of predicting whether a given author is or not the author of a given text), can be recast in terms of SAV; we have presented two original algorithms (*Lazy AA* and *Stacked AA*) that do this for both AA and AV.

In order to compare DV-based authorship identification methods with their counterparts based on "standard" vectors, we have carried out experiments on four datasets of texts labelled by author (one of which we have created ourselves and we here make publicly available for the first time) and representative of different textual genres, lengths, and styles, and on three authorship identification tasks (SAV, AA, AV). Our experiments have shown that DV-based methods are particularly suited to some authorship identification tasks and are not suited to others. For instance, the results indicate that neither standard methods nor DV-based methods clearly outperform each other on open-set SAV (see Section 4.4.2). Instead, DV-based methods vastly outperform the competition on three important tasks, i.e., (a) on closed-set SAV (see Section 4.4.1), (b) on closed-set AA (see Section 4.5), and (c) on AV (see Section 4.5). As we have argued, these benefits derive from the fact that, in many cases, DV-based methods may exploit more training data than methods based on standard vectors (see Section 2.3), and that DVs may make training more robust also when the above is not the case (see Section 2.4).

In future work, we would like to study "diff-functions" other than the absolute difference of (a static, fixed increasing function of) the feature frequencies of the two documents, by testing the possibility of dynamically *learning* such functions from data, in the style of [34]. Other aspects worth exploring include testing DVs in authorship profiling tasks, such as native language identification.

REFERENCES

[1] Charu C. Aggarwal. 2014. Instance-based learning: A survey. In *Data Classification: Algorithms and Applications.* Charu C. Aggarwal (Ed.), CRC Press, London, UK, 157–185.

[2] Shlomo Argamon and Patrick Juola. 2011. Overview of the international authorship identification competition at PAN 2011. In *Proceedings of the Working Notes of the 2011 Conference and Labs of the Evaluation Forum (CLEF 2011).* Amsterdam, NL.

[3] Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2009. Automatically profiling the author of an anonymous text. *Communications of the ACM* 52, 2 (2009), 119–123. DOI: https://doi.org/10.1145/1461928.1461959

[4] Alberto Bartoli, Alex Dagri, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2015. An author verification approach based on differential features. In *Proceedings of the Working Notes of the 2015 Conference and Labs of the Evaluation Forum (CLEF 2015)*. Toulouse, FR.

[5] Dario Benedetto, Mirko Degli Esposti, and Giulio Maspero. 2013. The puzzle of basil's epistula 38: A mathematical approach to a philological problem. *Journal of Quantitative Linguistics* 20, 4 (2013), 267–287. DOI: https://doi.org/10.1080/09296174.2013.830549

[6] Janek Bevendorff, Matthias Hagen, Benno Stein, and Martin Potthast. 2019. Bias analysis and mitigation in the evaluation of authorship verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*. Firenze, IT, 6301–6306.

[7] Carole E. Chaski. 2005. Who's at the keyboard? Authorship attribution in digital evidence investigations. *International Journal of Digital Evidence* 4, 1 (2005).

[8] Silvia Corbara, Alejandro Moreo, and Fabrizio Sebastiani. 2023. Syllabic quantity patterns as rhythmic features for latin authorship attribution. *Journal of the Association for Information Science and Technology* 74, 1 (2023), 128–141. DOI: https://doi.org/10.1002/asi.24660

[9] Silvia Corbara, Alejandro Moreo, Fabrizio Sebastiani, and Mirko Tavoni. 2019. The epistle to cangrande through the lens of computational authorship verification. In *Proceedings of the 1st International Workshop on Pattern Recognition for Cultural Heritage (PatReCH 2019)*. Trento, IT, 148–158. DOI: https://doi.org/10.1007/978-3-030-30754-7_15

[10] Silvia Corbara, Alejandro Moreo, Fabrizio Sebastiani, and Mirko Tavoni. 2022. MedLatinEpi and MedLatinLit: Two datasets for the computational authorship analysis of medieval latin texts. *ACM Journal of Computing and Cultural Heritage* 15, 3 (2022), 57:1–57:15. DOI: https://doi.org/10.1145/3485822

[11] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. 2003. Authorship attribution with support vector machines. *Applied Intelligence* 19, 1/2 (2003), 109–123.

[12] Pedro M. Domingos and Michael J. Pazzani. 1996. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*. Bari, IT, 105–112.

[13] Maciej Eder. 2011. Style-markers in authorship attribution: A cross-language study of the authorial fingerprint. *Studies in Polish Linguistics* 6, 1 (2011), 99–114.

[14] Peter A. Flach. 2017. Classifier calibration. In *Encyclopedia of Machine Learning* (2nd ed.), Claude Sammut and Geoffrey I. Webb (Eds.), Springer, 212–219.

[15] Christopher W. Forstall, Sarah L. Jacobson, and Walter J. Scheirer. 2011. Evidence of intertextuality: Investigating paul the deacon's angustae vitae. *Literary and Linguistic Computing* 26, 3 (2011), 285–296.

[16] Tim Gollub, Martin Potthast, Anna Beyer, Matthias Busse, Francisco M. Rangel Pardo, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2013. Recent trends in digital text forensics and its evaluation: Plagiarism detection, author identification, and author profiling. In *Proceedings of the 4th International Conference of the CLEF Initiative (CLEF 2013)*. Valencia, ES, 282–302. DOI: https://doi.org/10.1007/978-3-642-40802-1_28

[17] Abdulmecit Gungor. 2018. *Benchmarking Authorship Attribution Techniques using over a Thousand Books by Fifty Victorian Era Novelists*. Master's thesis. Department of Computer and Information Science, Purdue University, Indianapolis, US.

[18] Marjan Hosseinia and Arjun Mukherjee. 2018. Experiments with neural networks for small and large scale authorship verification. arXiv:1803.06456. Retrieved from https://arxiv.org/abs/1803.06456

[19] Catherine Ikae. 2021. UniNE at PAN-CLEF 2021: Authorship verification. In *Proceedings of the Working Notes of the 2021 Conference and Labs of the Evaluation Forum (CLEF 2021)*. Bucharest, RO, 1995–2003.

[20] Patrick Juola. 2006. Authorship attribution. *Foundations and Trends in Information Retrieval* 1, 3 (2006), 233–334. DOI: https://doi.org/10.1561/1500000005

[21] Jakub Kabala. 2020. Computational authorship attribution in medieval Latin corpora: The case of the monk of lido (ca. 1101–08) and gallus anonymous (ca. 1113–17). *Language Resources and Evaluation* 54, 1 (2020), 25–56. DOI: https://doi.org/10.1007/s10579-018-9424-0

[22] Mike Kestemont, Enrique Manjavacas, Ilia Markov, Janek Bevendorff, Matti Wiegmann, Efstathios Stamatatos, Benno Stein, and Martin Potthast. 2021. Overview of the cross-domain authorship verification task at PAN 2021. In *Proceedings of the Working Notes of the 2021 Conference and Labs of the Evaluation Forum (CLEF 2021)*. Bucharest, RO, 1743–1759.

[23] Mike Kestemont, Sara Moens, and Jeroen Deploige. 2015. Collaborative authorship in the twelfth century: A stylometric study of hildegard of bingen and guibert of gembloux. *Digital Scholarship in the Humanities* 30, 2 (2015), 199–224. DOI: https://doi.org/10.1093/llc/fqt063

[24] Mike Kestemont, Efstathios Stamatatos, Enrique Manjavacas, Walter Daelemans, Martin Potthast, and Benno Stein. 2019. Overview of the cross-domain authorship attribution task at PAN-2019. In *Proceedings of the Working Notes of the 2019 Conference and Labs of the Evaluation Forum (CLEF 2019)*. Lugano, CH, 1–15.

[25] Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. 2018. Overview of the author identification task at PAN-2018: Cross-domain authorship attribution and style change detection. In *Proceedings of the Working Notes of the 2018 Conference and Labs of the Evaluation Forum (CLEF 2018)*. Avignon, FR, 1–25.

[26] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *Proceedings of the 15th European Conference on Machine Learning (ECML 2004)*. Pisa, IT, 217–226. DOI: https://doi.org/10.1007/978-3-540-30115-8_22

[27] Moshe Koppel, Shlomo Argamon, and Anat R. Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing* 17, 4 (2002), 401–412. DOI: https://doi.org/10.1093/llc/17.4.401

[28] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology* 60, 1 (2009), 9–26. DOI: https://doi.org/10.1002/asi.20961

[29] Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology* 65, 1 (2014), 178–187. DOI: https://doi.org/10.1002/asi.22954

[30] Samuel Larner. 2014. *Forensic Authorship Analysis and the World Wide Web*. Springer.

[31] Antonio Menta and Ana Garcia-Serrano. 2021. Authorship verification with neural networks via stylometric feature concatenation. In *Proceedings of the Working Notes of the 2021 Conference and Labs of the Evaluation Forum (CLEF 2021)*. Bucharest, RO.

[32] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Lightweight random indexing for polylingual text classification. *Journal of Artificial Intelligence Research* 57 (2016), 151–185. DOI: https://doi.org/10.1613/jair.5194

[33] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2018. Revisiting distributional correspondence indexing: A Python reimplementation and new experiments. arXiv:1810.09311. Retrieved from https://arxiv.org/abs/1810.09311

[34] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2020. Learning to weight for text classification. *IEEE Transactions on Knowledge and Data Engineering* 32, 2 (2020), 302–316. DOI: https://doi.org/10.1109/TKDE.2018.2883446

[35] Frederick Mosteller and David L. Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, Reading, MA.

[36] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2005)*. Arlington, US, 413–420.

[37] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*. Bonn, DE, 625–632. DOI: https://doi.org/10.1145/1102351.1102430

[38] John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*. Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.), The MIT Press, Cambridge, MA, 61–74.

[39] Anderson Rocha, Walter J. Scheirer, Christopher W. Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne Carvalho, and Efstathios Stamatatos. 2017. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security* 12, 1 (2017), 5–33. DOI: https://doi.org/10.1109/TIFS.2016.2603960

[40] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24, 5 (1988), 513–523.

[41] Jacques Savoy. 2019. Authorship of pauline epistles revisited. *Journal of the Association for Information Science and Technology* 70, 10 (2019), 1089–1097. DOI: https://doi.org/10.1002/asi.24176

[42] Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Computational Linguistics* 40, 2 (2014), 269–310. DOI: https://doi.org/10.1162/COLI_a_00173

[43] Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology* 60, 3 (2009), 538–556. DOI: https://doi.org/10.1002/asi.21001

[44] Efstathios Stamatatos. 2016. Authorship verification: A review of recent advances. *Research in Computing Science* 123 (2016), 9–25.

[45] Joel R. Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*. Mumbai, IN, 2585–2602.

[46] Enrico Tuccinardi. 2017. An application of a profile-based method for authorship verification: Investigating the authenticity of pliny the younger's letter to trajan concerning the christians. *Digital Scholarship in the Humanities* 32, 2 (2017), 435–447. DOI: https://doi.org/10.1093/llc/fqw001

[47] Raija Vainio, Reima Välimäki, Anni Hella, Marjo Kaartinen, Teemu Immonen, Aleksi Vesanto, and Filip Ginter. 2019. Reconsidering authorship in the ciceronian corpus through computational authorship attribution. *Ciceroniana On Line* 3, 1 (2019), 15–48. DOI: https://doi.org/10.13135/2532-5353/3518

[48] Janith Weerasinghe, Rhia Singh, and Rachel Greenstadt. 2021. Feature vector difference based authorship verification for open world settings. In *Proceedings of the Working Notes of the 2021 Conference and Labs of the Evaluation Forum (CLEF 2021)*. Bucharest, RO.

[49] David H. Wolpert. 1992. Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259. DOI: https://doi.org/10.1016/s0893-6080(05)80023-1

[50] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research* 5 (2004), 975–1005.

[51] Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML 1997)*. Nashville, US, 412–420.

[52] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. Edmonton, CA, 694–699. DOI: https://doi.org/10.1145/775107.775151

[53] Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technologies* 57, 3 (2006), 378–393.

Received 23 January 2023; revised 3 July 2023; accepted 10 July 2023