Received June 24, 2021, accepted July 12, 2021, date of publication July 19, 2021, date of current version July 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3098192

Authorship Identification of Electronic Texts

MAHMOUD KHONJI[1], (Member, IEEE), YOUSSEF IRAQI[1,2], (Senior Member, IEEE), AND LOUBNA MEKOUAR[2], (Senior Member, IEEE)

[1]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
[2]School of Computer Science, Mohammed VI Polytechnic University, Ben Guerir 43150, Morocco

Corresponding author: Youssef Iraqi (youssef.iraqi@ieee.org)

ABSTRACT Electronic text stylometry is concerned with analyzing the writing styles of input electronic texts to extract information about their authors. For example, such extracted data could be the authors' identity or other aspects, such as their gender and age group. This survey paper presents the following contributions: 1) A description of all stylometry problems in probability terms, under a unified notation. 2) A survey of data representation (or feature extraction) methods. 3) A comprehensive evaluation of 23, 760 feature extraction methods followed by a thorough discussion of the results. This extensive evaluation is critical since the known data representation methods are often not evaluated under the same unified testbed.

INDEX TERMS Author identification, author verification, authorship analysis, text analysis, text categorization, forensics, natural language processing, feature extraction, supervised/unsupervised learning, classification.

I. INTRODUCTION

Improving solvers of stylometry problems is essential for enhancing various application domains, such as forensics, privacy (or anti-forensics), active-authentication [1]–[3], the detection of compromised accounts [4], recommender systems [5], deception detection, market analysis, and medical diagnosis [6], [7]. Author identification can also be accurately performed on program source codes [8], [9] as well as compiled binaries [10]. Enhancing such application domains is growing increasingly more interesting thanks to the availability of large amounts of textual data via the Internet.

Fundamentally, electronic text stylometry problems aim at inferring information about authors of input electronic texts. Such inferred information could be the identity of the authors, their genders, age groups, personality types, or even the diagnosis of specific illnesses [6], [7], [11]–[15]. A common taxonomy of electronic text stylometry problem solvers that is often followed by the literature is as follows:

- Author Attribution (AA): given a set of texts whose authors are known beforehand, find a classification model that predicts which of the known authors is also the author of the input test texts whose authors are not known. The target classification label, in this case, is the identity of the author.
- Author Clustering (AC): given a set of texts whose authors are not known, cluster the texts such that each cluster only contains texts written by only one author. The target classification label, in this case, is cluster identifiers.
- Author Verification (AV): given a pair of texts (or a pair of text sets such that texts within each set are written by one author) predict whether the texts are written by the same author. The target classification label, in this case, is either "*yes, the first text is written by the same author as the second text*" or "*no, the first text is written by someone else other than that of the second text.*"
- Author Profiling (AP): given a set of texts, identify the profile attributes of its author (regardless of who its author is). Examples of profile attributes are gender (i.e., male or female) and age group (e.g., 10s, 20s, and 30s). The target classification label is "*male*" or "*female* in the case of gender detection, and "*10s*", "*20s*", and "*30s*" in the case of age-group detection.
- Author Diarization (AD): given a single text whose authors are unknown, cluster its parts such that each cluster only has parts written by a single author. The target classification label, in this case, is cluster identifiers.

The same list of stylometry problem categories could also be presented from the perspective of the information that is intended to be inferred and the problem assumptions as follows:

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

TABLE 1. Contingency table of stylometry problems.

		Target information to infer	
		Author's identity	Author's profile attributes
Problem assumption	Open-set	AC, AV, AD	
	Closed-set	AA	AP

- Target information: authors' identities (AA, AC, AD, AV) or their profile attributes (AP).
- Problem assumptions: Is it assumed that the classification model is expected to handle situations where the actual author of the input test text is not represented in the learning set? If so, the problem is referred to as an *open-set* problem. Otherwise, the problem is referred to as a *closed-set* problem.
  For example, AV problems expect their solvers to tell whether the first text is written by someone other than the known author. This expectation increases the difficulty of the problem as the model is not only expected to model authors represented in the learning set but also any other author.
  Other open-set problems are AC and AD as they are expected to handle the situation where a text, or a text part, is written by someone else other than those of the known texts or text parts.
  On the other hand, AA and AP are strictly closed-set problems since AA expects that the correct author label to a given test text to be that of one of the known texts in the learning set. Similarly, the AP problem (based on the current state of the literature) assumes that the target profile attribute is necessarily one that is known in the learning set.

The significance of enhancing solvers of AV problems, relative to other stylometry problems, can be further appreciated thanks to their following properties:

- AV problems are known as the fundamental problem of stylometry because the other stylometry problems can be decomposed into a set of AV problems [6], [16]–[19].
- Due to the open-set nature of AV problems and their solvers, they have a broader application domain than the closed-set stylometry problems.

In other words, enhancing the performance of AV problem solvers is highly important when the objective is identifying authors in realistic problem domains where input test texts may be written by previously unseen authors [19]–[22].

Table 1 summarizes the categories of all stylometry problems from the perspective of the target information that they seek to infer and their problem assumptions.

While this paper focuses on stylometric methods for identifying authors of electronic texts, it might be important to draw attention to the fact that non-stylometric methods also exist. For example, if sufficient access is obtained on the messaging infrastructure (e.g., the network) that an author used to publish their works, one can use deterministic methods to track the text's source, ultimately leading to its author. Gaining access to such infrastructure can be achieved by having legitimate administrative privileges or illegitimately using malware, back-doors, or compromised network tunnels. What sets the stylometric methods apart from non-stylometric ones is that the former can be executed without requiring access to the underlying messaging infrastructure. Because of this, stylometric author identification methods can be applicable in cases where the non-stylometric methods fail to apply, such as when obtaining adequate access to the messaging infrastructures is not feasible. Additionally, in scenarios where author identification problems are solvable by stylometric and non-stylometric methods, the stylometric techniques can still be used to provide further evidence to increase confidence in the solution to the author identification problem.

The focus of this paper is to survey author identification methods. Due to the diverse nature of the problems, and the uncertainty associated with the relative contribution of many of the feature extraction methods, we found it useful to include the following:

- An objective definition of the fundamental stylometry problems in probability terms using a common notation.
- A generalized definition of many of the feature extraction methods and evaluating their implementation in a common testing bed. To the best of our knowledge, this is the first time such diverse feature extraction methods are assessed under the same testing bed. This gives us a closer look at how they relate to each other in terms of their performance.

This paper is structured as follows: Section II presents some of the most significant challenges that face today's state of electronic text stylometry. Section III defines several fundamental classification problems that can be used to model all stylometry problem solvers. Section IV presents a comprehensive definition of text representation methods, as commonly used in the literature, in addition to our generalizations. Section V defines some of the stylometry problems and introduces several notable stylometry problem solvers. Section VI discusses the critical drawbacks of current methods. Section VII presents our evaluation methodology of the extensive set of feature extraction functions, followed by the evaluation results in Section VIII. Section IX offers information about evaluation reproducibility. Section X presents discussions on possible future research directions. Section XI draws the concluding remarks.

II. CHALLENGES

Some of the most significant challenges that face electronic text stylometry problems are identifying and enhancing existing methods' classification accuracy. Namely:

- Optimization of algorithms: most of the proposed stylometry algorithms, including state-of-the-art methods, often contain parameters that are, at least, not adequately discussed or evaluated. This may naively reduce the space of parameters, which restricts our ability to achieve more accurate stylometry problem solvers.

Therefore, it is critical to question the various aspects of state-of-the-art stylometry problem solver methods to identify such parameters and alternative variants.

- Cross-domain stylometry: author identification problem solvers tend to classify a pair of texts written under distinct domains (e.g., distinct topics or genres) to be written by different authors, even when they are not. Similarly, texts that fall under the same domain are more likely to be classified to be written by the same author due to their domain similarity, even when they are not. This issue is a critical limitation of accurately solving AV problems in Big Data scenarios where cross-domain texts are not uncommon.
- Generalization of existing data representation techniques: many methods of representing electronic texts (or feature extraction methods) are proposed; however, the current state of the literature on stylometry lacks good generalizations for such methods. This may effectively reduce our ability to identify novel variants of existing stylometry methods and feature extraction functions.
- Software availability: There is a lack of available and extensive software that implements the many existing stylometry methods and feature extraction functions. There is often a tremendous need to re-develop the many proposed techniques or procedures. Because of the sheer amount of effort required to develop such functions, it is expected that most of the methods are not adequately evaluated. As a result, the actual value of the numerous independent contributions relative to each other is often not sufficiently known.

This survey paper moves towards addressing the last two challenges, namely: generalizing existing data representation methods and releasing our extensive feature extraction library under a premising open-source license.

III. FUNDAMENTAL CLASSIFICATION PROBLEMS

This section introduces fundamental classification problems and their solvers that are relevant to solving all stylometry problems.

The literature tackles the following fundamental stylometry problems: Single-domain Closed-set Classification (SCC), Single-domain Open-set Classification (SOC), Multi-domain Closed-set Classification (MCC), or Multi-domain Open-set Classification (MOC) problems. These are detailed in this section. Figure 1 visualizes this taxonomy. In reality, it is possible for hybrid problems to exist, for which we give examples in later sections.

The following sections will present formal definitions of SCC, MCC, SOC, and MOC.

A. NOTATION

This section defines the notation that will be followed throughout this paper.

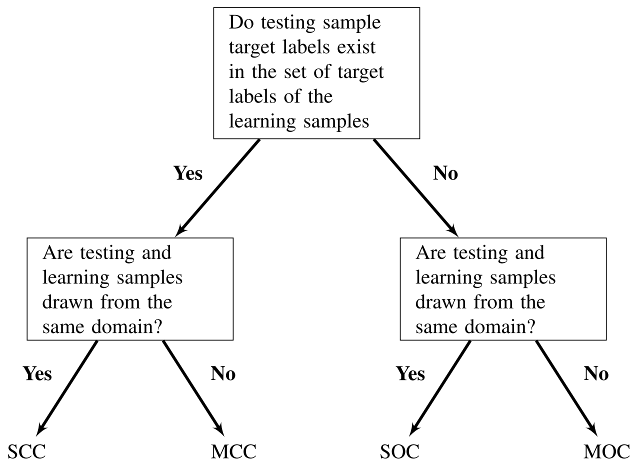
Do testing sample target labels exist in the set of target labels of the learning samples
Yes
No
Are testing and learning samples drawn from the same domain?
Are testing and learning samples drawn from the same domain?
Yes
No
Yes
No
SCC
MCC
SOC
MOC

FIGURE 1. A categorization of fundamental stylometry problems.

- Let $\mathcal{I}$ be the index set of all texts, $\mathcal{I}_L \subset \mathcal{I}$ be that of the learning samples, and $\mathcal{I}_T \subseteq \mathcal{I} \setminus \mathcal{I}_L$ be that of the testing samples.
- Let $\mathcal{D}$ be the index set of all classification domains (e.g., topics and genres).
- Let $\mathcal{Q}$ be the index set of all classification tasks (e.g., author identification, gender identification, and age-group identification).
- For any $(i, d, q) \in \mathcal{I} \times \mathcal{D} \times \mathcal{Q}$:
  - $x_{i,d}$ is a text that is written in domain d. A text's domain could be defined based on its topic, genre, or time of authorship.
  - $\mathbf{x}_{i,d}$ is the vector-representation of the text $x_{i,d}$. For example, if a text $x_{i,d}$ to be represented based on the frequency of patterns (e.g., sequences of words), then each component of the vector $\mathbf{x}_{i,d}$ represents the frequency of a specific pattern.
  - $y_{i,q}$ is the classification label of text x_i under task q. For example, if the task is AA, AV, or AC, the labels represent author identifiers. On the other hand, if the task is AP for gender detection, then $y_{i,q} \in \{\text{male, female}\}$.
- fex is a function that represents texts as dim dimensional vectors in $\mathbb{R}^{\text{dim}}$:

$$\text{fex} : \{x_{i,d} : i \in \mathcal{I}, d \in \mathcal{D}\} \to \mathbb{R}^{\text{dim}} x_{i,d} \mapsto \mathbf{x}_{i,d} \quad (1)$$

- $\mathcal{X}_d = \{\mathbf{x}_{i,d} : i \in \mathcal{I}\}$ is the set of all samples in domain d, $\mathcal{X}_{d,L} = \{\mathbf{x}_{i,d} : i \in \mathcal{I}_L\}$ is that of the learning set, $\mathcal{X}_{d,T} = \{\mathbf{x}_{i,d} : i \in \mathcal{I}_T\}$ is that of the testing set, and $\mathcal{X}_{d,y_{i,q}} = \{\mathbf{x}_{j,d} : j \in \mathcal{I}, y_{j,d} = y_{i,d}\}$ is the set of all samples that are associated with the classification label $y_{i,q}$ in domain d. Additionally, $X_d, X_{d,L}, X_{d,T}$ and $X_{d,y_{i,q}}$ are random variables that take values in sets $\mathcal{X}_d$, $\mathcal{X}_{d,L}$, $\mathcal{X}_{d,T}$ and $\mathcal{X}_{d,y_{i,q}}$, respectively.
- For any classification task $q \in \mathcal{Q}$:
  - $\mathcal{Y}_q = \{y_{i,q} : i \in \mathcal{I}\}$ is the set of labels of the samples under classification task q, and Y_q is a random variable that takes values in $\mathcal{Y}_q$.

- $\mathcal{Y}_{L,q} = \{y_{i,q} : i \in \mathcal{I}_L\}$ is the set of labels of the learning samples under classification task q, and $Y_{L,q}$ is a random variable that takes values in $\mathcal{Y}_{L,q}$.
- $\mathcal{Y}_{T,q} = \{y_{i,q} : i \in \mathcal{I}_T\}$ is that of the testing samples, and $Y_{T,q}$ is a random variable that takes values in $\mathcal{Y}_{T,q}$.

- $z_{b\to d}$ is a Domain Adaptation (DA) function that transforms represented texts in domain b into their expected representation in domain d, which estimates their value should they have been written by a process with the same classification label. More formally, let $\mathcal{Z}_{b\to d} = \{z_{b\to d}(\mathbf{x}_{i,b}) : \mathbf{x}_{i,b} \in \mathcal{X}_b\}$, and $Z_{b\to d}$ be a random variable that takes values in the set $\mathcal{Z}_{b\to d}$, such that the following joint probability density functions (PDFs) are equivalent:

$$f_{Z_{b\to d},Y_q} = f_{X_d,Y_q} \tag{2}$$

B. SINGLE-DOMAIN CLOSED-SET CLASSIFICATION (SCC)

For any classification task $q \in \mathcal{Q}$, any domain $d \in \mathcal{D}$, and for any vector-represented testing text $\mathbf{x}_{i,d} \in \mathcal{X}_{d,T}$, classification models aim to predict $y_{i,q}$ by finding the prediction $\hat{y}_{i,q}$ as follows:

$$\hat{y}_{i,q} = \operatorname*{arg\,max}_{y\in\mathcal{Y}_{T,q}} \Pr(Y_{T,q} = y | X_{d,T} = \mathbf{x}_{i,d}) \tag{3}$$

However, what identifies a classification model as an SCC is its input and its assumptions that are used to estimate the probabilities in (3) as listed below.

Input 1: Target classification task is q, for some $q \in \mathcal{Q}$.

Input 2: The set of learning samples $\mathcal{X}_{d,L}$ from domain d.

Input 3: For any learning sample $\mathbf{x}_{i,d} \in \mathcal{X}_{d,L}$, its corresponding classification label $y_{i,q}$ under task q is given as input.

Input 4: The set of testing samples $\mathcal{X}_{d,T}$ under the domain d.

Assumption 1: For any input sample $\mathbf{x}_{i,d} \in \mathcal{X}_{d,T}$, its classification label $y_{i,q} \in \mathcal{Y}_{L,q}$. In other words, $\mathcal{Y}_{T,q} \subseteq \mathcal{Y}_{L,q}$.

Assumption 2: All input samples of the learning set belong to the same domain d as those of the testing set.

Assumption 1 signifies that, for any test text, there exists a sample text in the learning set that has the same classification label as that of the test text.

Assumption 2 signifies that all learning and testing samples belong to only one domain.

Therefore, for any task $q \in \mathcal{Q}$, any label $y \in \mathcal{Y}_{T,q}$, and any represented text $\mathbf{x}_{i,d} \in \mathcal{X}_T$, Assumptions 1 and 2 imply that:

$$\Pr(Y_{T,q} = y | X_{d,T} = \mathbf{x}_{i,d}) = \Pr(Y_{L,q} = y | X_{d,L} = \mathbf{x}_{i,d}) + \epsilon \tag{4}$$

where ϵ is an irreducible error term. This means that by using the learning set, we can estimate $\Pr(Y_{T,q} = y | X_{d,T} = \mathbf{x}_{i,d})$ and use it as a reliable estimator for $\Pr(Y_{L,q} = y | X_{d,L} = \mathbf{x}_{i,d})$ as follows:

$$\begin{aligned}\hat{y}_{i,q} &= \operatorname*{arg\,max}_{y\in\mathcal{Y}_{T,q}} \Pr(Y_{T,q} = y | X_{b,T} = \mathbf{x}_{i,b}) \\ &= \operatorname*{arg\,max}_{y\in\mathcal{Y}_{L,q}} \Pr(Y_{L,q} = y \big| X_{d,L} = \mathbf{x}_{i,b}) + \epsilon \end{aligned} \tag{5}$$

C. MULTI-DOMAIN CLOSED-SET CLASSIFICATION (MCC)

MCC problem solvers are identical to those of SCC except for dropping Assumption 2. In other words, the testing set $\mathcal{X}_{b,T}$ falls under domain b, where $b \neq d$ (recall that the learning set is $\mathcal{X}_{d,L}$ which falls under domain d). Due to this, the following assumption:

$$\Pr(Y_{T,q} = y | X_{b,T} = \mathbf{x}_{i,b}) = \Pr(Y_{L,q} = y | X_{d,L} = \mathbf{x}_{i,b}) + \epsilon_{b,d}$$

often results in an error term $\epsilon_{b,d}$ that is too large. Our evaluations indicate that the error can be large enough to degrade the classification accuracy down to random chance guessing.

To address this problem, MCC assumes the following:

Assumption 3: There exists function $z_{b\to d}$ such that, for any $\mathbf{x}_{i,b} \in \mathcal{X}_{b,T}$ and any $y \in \mathcal{Y}_{T,q}$,

$$\begin{aligned}\Pr(Y_{T,q} &= y | X_{b,T} = \mathbf{x}_{i,b}) \\ &= \Pr\left(Y_{L,q} = y \big| X_{d,L} = z_{b\to d}(\mathbf{x}_{i,b})\right) + \epsilon_{z_{b\to d}}\end{aligned}$$

where $\epsilon_{z_{b\to d}} < \epsilon_{b,d}$.

Therefore, MCC problem solvers can be modeled as follows:

$$\begin{aligned}\hat{y}_{i,q} &= \operatorname*{arg\,max}_{y\in\mathcal{Y}_{T,q}} \Pr(Y_{T,q} = y | X_{b,T} = \mathbf{x}_{i,b}) \\ &= \operatorname*{arg\,max}_{y\in\mathcal{Y}_{L,q}} \Pr\left(Y_{L,q} = y \big| X_{d,L} = z_{b\to d}(\mathbf{x}_{i,b})\right) + \epsilon_{z_{b\to d}}\end{aligned} \tag{6}$$

The Hybrid SCC-MCC problem solvers can be described by modifying the probability:

$$\Pr\left(Y_{L,q} = y \big| X_{d,L} = z_{b\to d}(\mathbf{x}_{i,b})\right)$$

to be conditioned on more events. For example, if the training set contains three learning sets, two of which are from distinct domains d_1 and d_2, then the following probability can be used instead:

$$\Pr\left(Y_{L,q} = y \left| \begin{cases} X_{b,L} = \mathbf{x}_{i,b}, \\ X_{d_1,L} = z_{b\to d_1}(\mathbf{x}_{i,b}), \\ X_{d_2,L} = z_{b\to d_2}(\mathbf{x}_{i,b}) \end{cases} \right.\right).$$

The same concept can be applied to other probability terms to describe hybrids of different problem types, such as SOC and MOC. However, we omit such details for brevity.

D. SINGLE-DOMAIN OPEN-SET CLASSIFICATION (SOC)

Similar to SCC, SOC problems also take Inputs 1, 2, 3, and 4, and also follow Assumption 2. However, the SOC problems differ from SCC problems in that the former do not seek to identify the actual labels, but rather to test whether a pair of sets (possibly, each composed of a single represented text) share the same target classification label under the same task (regardless of the values of such labels).

Since all represented texts in this section fall under the same domain d, and since all classification labels belong to the same classification task q, domain and classification task indices will be omitted for the sake of simplifying the notation in this section.

Formally, let $\mathcal{X}_{y_i,1} \subseteq \mathcal{X}_{y_i}$, and $\mathcal{X}_{y_j,2} \subseteq \mathcal{X}_{y_j}$ be two subsets that contain represented texts that correspond to classification labels y_i and y_j, respectively, such that $\mathcal{X}_{y_i,1} \cup \mathcal{X}_{y_j,2} \in \mathcal{X}_T$, and $\mathcal{X}_{y_i,1} \cap \mathcal{X}_{y_j,2} = \emptyset$ (to avoid the possibility of the existence of trivial solutions by simply finding identical represented texts). Then, the SOC problem at hand is to infer whether $y_i = y_j$. This can be answered in probability terms as follows:

$$\begin{cases} \text{Yes, } y_i = y_j & \text{if } \Pr(Y_{T,1} = Y_{T,2} | X_{T,1} = \mathcal{X}_{y_i,1}, \\ & \qquad X_{T,2} = \mathcal{X}_{y_j,2}) > t \\ \text{No, } y_i \neq y_j & \text{otherwise} \end{cases} \tag{7}$$

where $Y_{T,1}$ and $Y_{T,2}$ are independent random variables that take values in $\mathcal{Y}_T$, $X_{T,1}$, and $X_{T,2}$ are independent random variables that take values in $\mathcal{X}_T$, and t is a threshold. If the objective is to maximize the classification accuracy, then $t = 0.5$ is optimum.

Similar to the SCC problems, we estimate the probability in (7) by analyzing the learning samples and their corresponding labels, with the assumption that this probability is equivalent to the following probability:

$$\Pr(Y_1 = Y_2 | X_{L,1} = \mathcal{X}_{y_i,1}, X_{L,2} = \mathcal{X}_{y_j,2}) + \epsilon \tag{8}$$

where Y_1 and Y_2 are independent random variables that take values in $\mathcal{Y}$, and $X_{L,1}$ and $X_{L,2}$ are random variables that take values in $\mathcal{X}_L$, such that ϵ is adequately small. Therefore, (7) can be estimated as follows:

$$\begin{cases} \text{Yes, } y_i = y_j & \text{if } \Pr(Y_1 = Y_2 | X_{L,1} = \mathcal{X}_{y_i,1}, \\ & \qquad X_{L,2} = \mathcal{X}_{y_j,2}) > 0.5 \\ \text{No, } y_i \neq y_j & \text{otherwise} \end{cases} \tag{9}$$

However, since SOC problems do not assume that $\mathcal{Y}_T \subseteq \mathcal{Y}_L$, it is important to ensure that, when the probability function in (9) is being estimated, the probability function only identifies what generally makes represented texts of distinct labels differ from each other, without being too specific to labels of the learning set.

In fact, it is common for SOC evaluation datasets (e.g., such as those of PAN [23]–[25]) to strictly define $\mathcal{Y}_T \cap \mathcal{Y}_L = \emptyset$. This ensures that SOC models are not rewarded for being SCC models that simply generalize for specific labels of the learning set, as opposed to generalizing for any label, including those unseen in $\mathcal{Y}_L$).

To demonstrate how to estimate the probability in (9), such that it generalizes to problems of the testing set, and without over-fitting samples of the learning set, consider the following hypothetical example of four subsets of represented texts that are obtained from the learning set $\mathcal{X}_{y_1,1} \subseteq \mathcal{X}_{y_1}$, $\mathcal{X}_{y_2,2} \subseteq \mathcal{X}_{y_2}$, $\mathcal{X}_{y_3,3} \subseteq \mathcal{X}_{y_3}$, and $\mathcal{X}_{y_4,4} \subseteq \mathcal{X}_{y_4}$, where we know beforehand that $y_1 = y_2$, $y_3 = y_4$, but $y_1 \neq y_3$. Additionally, let $X_{y_1,1}$, $X_{y_2,2}$, $X_{y_3,3}$, and $X_{y_4,4}$ be random variables that take values in the subsets, respectively. Furthermore, suppose that their PDFs visualize as presented in Figure 2.

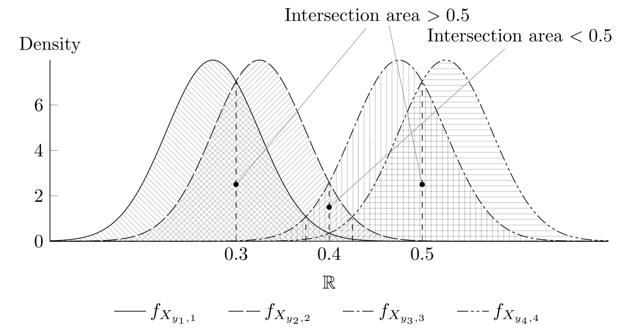

FIGURE 2. Hypothetical examples of the PDFs $f_{X_{y_1,1}}$, $f_{X_{y_2,2}}$, $f_{X_{y_3,3}}$ and $f_{X_{y_4,4}}$.

Then an example of an over-fitting generalization would be to state "*if the corresponding PDF of two subsets are centered nearby* 0.3 *or* 0.5*, then the pair share the same label, otherwise they do not*". Such generalizations over-fit the specific learning subsets in $\mathcal{X}_L$ as the subsets represent authors with their represented texts centered around 0.3 and 0.5.

On the other hand, a more robust generalization that is less likely to over-fit than the previous one is to state estimate the probability in (9) is by measuring the intersection area between the various PDF pairs as depicted in Figure 2. In this hypothetical example, subset pairs that share the same classification label, regardless of the value of the label, also share an intersection area greater than 0.5.

Worth noting that the SOC problems are often referred to in the literature as a fundamental problem of stylometry [6].

E. MULTI-DOMAIN OPEN-SET CLASSIFICATION (MOC)

MOC problem solvers are identical to those of the SOC except for further dropping one more assumption, namely Assumption 2. This significantly increases the difficulty of the solver, as texts of the learning set could be in a different domain than those of the testing set. As a result, the probability in (7) cannot be directly estimated from the probability in (9) as found from the learning set. This is because there is a domain mismatch between samples of the learning set and samples of the testing set.

Since all classification labels belong to the same classification task q, classification task indices will be omitted to simplify the notation in this section.

Formally, let $\mathcal{X}_{b,y_i,1} \subseteq \mathcal{X}_{b,y_i}$ and $\mathcal{X}_{b,y_j,2} \subseteq \mathcal{X}_{b,y_j}$ be two subsets that contain represented texts that correspond to classification labels y_i and y_j, respectively, such that $\mathcal{X}_{b,y_i,1} \cup \mathcal{X}_{b,y_j,2} \in \mathcal{X}_{b,T}$, and $\mathcal{X}_{b,y_i,1} \cap \mathcal{X}_{b,y_j,2} = \emptyset$. Then, similar to SOC problems, the MOC problem at hand is to infer whether $y_i = y_j$. However, unlike SOC problems, MOC problems are composed of represented texts that fall under domain b, where $b \neq d$ (recall that the learning set falls under domain d).

Therefore, estimating (7) by (9) would often result in an error term that is too large due to the domain mismatch between learning and testing sets. For example, does the intersection area threshold 0.5 that applies to domain d also applies to domain b?

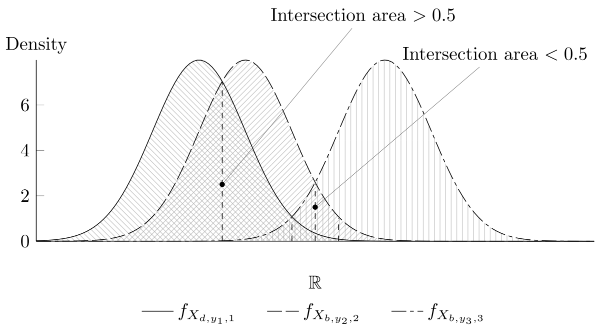

FIGURE 3. Hypothetical examples of the PDFs $f_{X_{d,y_1,1}}$, $f_{X_{b,y_2,2}}$ and $f_{X_{b,y_3,3}}$, before applying the DA function $z_{b\to d}$.

To extend the generalizations that are found from the learning set (e.g., the PDFs intersection area threshold), Assumption 3 is followed (similar to MCC) as follows:

$$\begin{cases} \text{Yes, } y_i = y_j & \text{if } \Pr(Y_{q,1} = Y_{q,2} | X_{d,L,1} = \mathcal{Z}_{b\to d,y_i,1}, \\ & \qquad X_{d,L,2} = \mathcal{Z}_{b\to d,y_j,2}) > 0.5 \\ \text{No, } y_i \neq y_j & \text{otherwise} \end{cases} \tag{10}$$

where $\mathcal{Z}_{b\to d,y_i,1} = \{z_{b\to d}(\mathbf{x}_{l,b}) : \mathbf{x}_{l,b} \in \mathcal{X}_{b,y_i,1}\}$ and $\mathcal{Z}_{b\to d,y_j,2} = \{z_{b\to d}(\mathbf{x}_{l,b}) : \mathbf{x}_{l,b} \in \mathcal{X}_{b,y_j,2}\}$.

To visualize this transformation by the DA function, consider the following hypothetical example of the following subsets of represented texts $\mathcal{X}_{d,y_1,1} \subseteq \mathcal{X}_{d,y_1}$ (note that its domain is d), $\mathcal{X}_{b,y_2,2} \subseteq \mathcal{X}_{b,y_2}$ and $\mathcal{X}_{b,y_3,3} \subseteq \mathcal{X}_{b,y_3}$ (note that the domain of the latter subsets is b), where we know beforehand that $y_1 = y_3$, but $y_1 \neq y_2$. Additionally, let $X_{d,y_1,1}$, $X_{b,y_2,2}$ and $X_{b,y_3,3}$ be random variables that take values in the subsets, respectively. Then, the intersection areas of the PDFs, before the DA transformation function $z_{b\to d}$ is applied, is depicted in the hypothetical example in Figure 3.

Note that in the example in Figure 3, the intersection area between the PDFs $f_{X_{d,y_i,1}}$ and $f_{X_{d,y_2,2}}$ is greater than 0.5, even though they do not share the same classification label (i.e., $y_1 \neq y_2$). On the other hand, the intersection area between the PDFs $f_{X_{d,y_i,1}}$ and $f_{X_{d,y_2,3}}$ is less than 0.5, although they share the same classification label (i.e., $y_1 = y_3$). This is possibly because the represented texts fall under distinct domains d and b.

On the other hand, Figure 4 depicts the PDFs of this example, except for applying the DA transformation function $z_{b\to d}$, accordingly. It can be seen from this example that once the transformation is used, the intersection area is greater than 0.5 between the PDFs that share the same classification label, while less than 0.5 when the intersecting PDFs do not share the same classification label.

In this hypothetical example, the implementation of the DA function $z_{b\to d}$ assumed that the effect of the domain variation from d to b is comprised of an increase in the mean and the variance of the PDFs. In other words, if the PDF $f_{X_{d,y_i}}$ follows the normal distribution $\mathcal{N}(\mu, \sigma^2)$, then $f_{X_{b,y_i}}$ follows the normal distribution $\mathcal{N}(\texttt{const}_1 + \mu, \texttt{const}_2 + \sigma^2)$.

As a result, for the DA transformation function, $z_{b\to d}$, to undo the effect of the domain variation, by representing

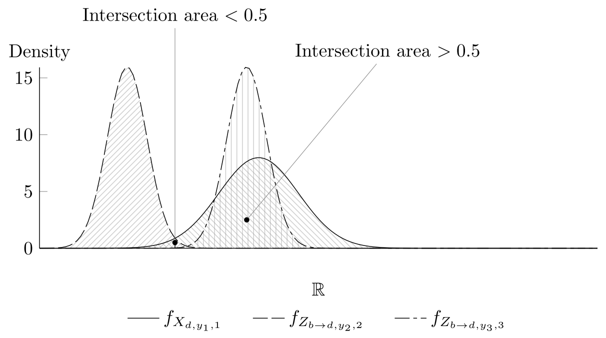

FIGURE 4. Hypothetical examples of the PDFs $f_{X_{d,y_1,1}}$, $f_{X_{b,y_2,2}}$ and $f_{X_{b,y_3,3}}$, after applying the DA function $z_{b\to d}$.

samples in the domain b in the domain d of the learning set, it adjusts the distribution of the samples in $\mathcal{X}_b$, such that their mean and variance are reduced by the constants $\texttt{const}_1$ and $\texttt{const}_2$, respectively.

The probabilistic models introduced in this section are used to define the stylometry problems objectively in probability terms. These models help in classifying any encountered stylometry problems. The solvers, on the other hand, are therefore estimations of these probability models. This estimation can be done either directly or indirectly. Indeed, while some solvers like NLP-based solvers may not directly attempt to estimate the probabilities or the density functions introduced in this section, such solvers indirectly imply the existence of such probabilities. Other solvers may make use of, say, kernel density estimation methods to find such density functions. The fact that a solver can offer answers to stylometry problems implies that such probabilities or density functions were indirectly defined.

IV. DATA REPRESENTATION METHODS

The following sections will present various feature extraction methods, such as richness-based and rewrite rules feature extraction methods, as well as n-gram-based ones. Frequency-based features (e.g., distribution of Part of Speech (POS) tags) are treated as special cases of n-grams for when $n = 1$. Additionally, other variants of n-grams, such as syntactic n-grams, are treated as special cases of our generalized view of n-gram-based methods, namely: the at least l-frequent $\texttt{dir}$-directed k-skipped n-grams. In other words, the use of dependency trees in syntactic n-grams is treated as a different direction for the sliding window of n-grams. In this context, classical n-grams assume that the direction is spatial.

A. VOCABULARY RICHNESS

For any text $x_{i,d}$, vocabulary richness measures [18] aim to quantify the vocabulary diversity of input text $x_{i,d}$ to solve stylometry problems. Examples of such measures are:

- Type-token ratio: the ratio of the total number of unique tokens to the total number of tokens:

$$\text{uniq}(N_{i,\text{tokens}})/N_{i,\text{tokens}} \tag{11}$$

where $N_{i,\text{tokens}}$ and $\text{uniq}(N_{i,\text{tokens}})$ are the total number of tokens and the total number of unique tokens in text $x_{i,d}$, respectively. A *token* is a general term that could refer, for example, to a word, a number, or a punctuation mark.

- Hapax legomena: the total number of words that occur once in $x_{i,d}$ which we denote by N_{i,words_1}
- Hapax dislegomena: the total number of words that occur twice in $x_{i,d}$ which we denote by N_{i,words_2}.

However, the measures above are sensitive to the size of $x_{i,d}$ (i.e., the scores change as a function of the length of text $x_{i,d}$). To minimize this, Yule's K [26] and Honore's R [27] are functions that aim to stabilize the measures, as defined below:

$$K_i = \frac{10^4(\sum_{w=1}^{\infty} w^2 N_{i,\text{words}_w} - N_{i,\text{tokens}})}{N_{i,\text{tokens}}^2} \tag{12}$$

where K_i is Yule's K measure for text $x_{i,d}$, and N_{words_w} is the total number of words in $x_{i,d}$ that occur exactly w many times.

$$R_i = \frac{100 \log(N_{i,\text{tokens}})}{1 - N_{i,\text{words}_1}/N_{i,\text{tokens}}} \tag{13}$$

where R_i is Honore's R measure for text $x_{i,d}$.

1) RELATION TO OUR NOTATION

For any text $x_{i,d}$, $\mathbf{x}_{i,d} = \text{fex}(x_{i,d})$ such that $\mathbf{x}_{i,d}[1]$ is a number that reflects type-token ratio, Hapax legomena, Hapax dislegomena, Yule's K, or Honore's R. In other words, $\mathbf{x}_{i,d}$ is a one dimensional real vector.

2) DISCUSSION

The underlying assumption of vocabulary richness-based feature extractions methods is that texts of identical labels generally tend to maintain sufficiently similar richness values. In contrast, texts of varying target classification labels tend typically to keep enough different richness values.

However, such assumption is often false as the richness methods are heavily and systematically dependent on the length of input texts, and Yule's K and Honore's R that attempt to stabilize them have questionable results [18]. For example, [28] shows that Yule's K is ineffective for identifying authors.

B. CLASSICAL n-GRAMS

The classical n-grams (or just n-grams as commonly referred to in the literature) is probably the most common data representation method applied in the literature of stylometry problems.

Only three parameters define the space of patterns that any implementation of classical n-grams can explore. The parameters are gram, n, and the length of the input text $x_{i,d}$, $\text{len}(x_{i,d})$, which are defined as follows:

Gram: this parameter defines the most fundamental unit of the processed text. For example, if grams are defined to be *words*, then the most basic unit of any text is considered to be words (i.e., strings of text that are separated by word separators, such as whitespace characters and punctuation marks). For any text $x_{i,d}$, let $x_{i,d}[g]$ be the g^{th} gram in text $x_{i,d}$. Below is a list of common definitions of grams in the literature:

- Characters: this definition of grams is among the most fundamental gram definitions by which input texts are considered to be constructed by arrays of characters. In this case, $x_{i,d}[g]$ denotes the g^{th} character in text $x_{i,d}$.
  Example text: suppose $x_{i,d}$ = "*Can I see? the boy said. Yes. Of course you can.*" (from *The Road* by Cormac McCarthy).
  Example grams: $x_{i,d}[1]$ = "C", $x_{i,d}[2]$ = "a", $x_{i,d}[3]$ = "n", $x_{i,d}[4]$ = " ", ..., $x_{i,d}[48]$ = ".", where " " represents a space.
- Letters: input texts are considered as arrays of letters. In this case, $x_{i,d}[g]$ denotes the g^{th} letter in text $x_{i,d}$.
  Example grams: $x_{i,d}[1]$ = "C", $x_{i,d}[2]$ = "a", $x_{i,d}[3]$ = "n", $x_{i,d}[4]$ = "I", ..., $x_{i,d}[34]$ = "n". Note that non-letter characters (e.g., whitespace characters and punctuation marks) are ignored.
- Punctuation marks: input texts are considered as arrays of punctuation marks, ignoring any other types of characters. In this case, $x_{i,d}[g]$ denotes the g^{th} punctuation mark in text $x_{i,d}$.
  Example grams: $x_{i,d}[1]$ = "?", $x_{i,d}[2]$ = ".", ..., $x_{i,d}[4]$ = ".".
- Words: input texts are considered as arrays of words, i.e., strings of characters that are separated by word separators.[1] In this case, $x_{i,d}[g]$ denotes the g^{th} word in text $x_{i,d}$.
  Example grams: $x_{i,d}[1]$ = "Can", $x_{i,d}[2]$ = "I", $x_{i,d}[3]$ = "see", ..., $x_{i,d}[11]$ = "can".
- Word shapes: input texts are considered as arrays of word shapes. Word shapes could be defined based on their characters cases (i.e., upper/lower cases) and type (e.g., letter/number) by which the word "*Apple*" has the shape "*Sssss*", and "*x86*" has the shape "*sDD*", where S, s, and D represent an upper case letter, a lower case letter, and a digit, respectively. In this case, $x_{i,d}[g]$ denotes the shape of the g^{th} word in text $x_{i,d}$.
  Example grams: $x_{i,d}[1]$ = "Sss", $x_{i,d}[2]$ = "S", $x_{i,d}[3]$ = "sss", ..., $x_{i,d}[11]$ = "sss".
- Function words: input texts are considered as arrays of function words, which are words that are used for grammatical proposes to join other words, such as "*and*", "*at*", and "*for*", ignoring any other types of words. In this case, $x_{i,d}[g]$ denotes the g^{th} function word in text $x_{i,d}$.
  Linguists traditionally identify function words on a per-language basis. Alternatively, since function words also happen to occur more frequently than *content words*, it is also possible to identify them heuristically by choosing the most frequent words in a given corpus (i.e., the most

[1]Examples of word separators are paragraph start, punctuation marks, and whitespace.

The	quick	fox	jumped	over	the	lazy	dog
DT	JJ	NN	VBD	IN	DT	JJ	NN

FIGURE 5. POS tags for the sentence "*the quick fox jumped over the lazy dog*" as identified by Stanford's statistical sentence parser, where DT, JJ, NN, VBD, and IN denote that the tagged word is a determiner, adjective, noun, past tense verb and preposition, respectively.

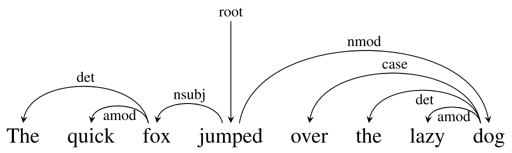

FIGURE 6. Dependency-based parse tree for the sentence "*the quick fox jumped over the lazy dog*" as identified by Stanford's statistical sentence parser.

frequent words in a corpus are likely to contain function words mostly).
Example grams: $x_{i,d}[1]$ = "Can", $x_{i,d}[2]$ = "I", $x_{i,d}[3]$ = "the", …, $x_{i,d}[5]$ = "can". Note that non-function words, such as "see" are ignored.

- POS tags: input texts are considered as arrays of word POS tags. In this case, $x_{i,d}[g]$ denotes the POS tag of the g^{th} word in text $x_{i,d}$. An example is presented in Figure 5.
Example grams: $x_{i,d}[1]$ = DT, $x_{i,d}[2]$ = JJ, $x_{i,d}[3]$ = NN, …, $x_{i,d}[8]$ = NN, where DT, JJ, and NN are POS tags of corresponding words as tagged by Stanford's sentence parser.[2] The tags are defined as per the Penn treebank.
- Dependency relation: input texts are considered as arrays of word dependency relation types. In this case, $x_{i,d}[g]$ denotes the dependency relation of the g^{th} word in text $x_{i,d}$ towards its parent as per the dependency-based parse tree of the sentence the g^{th} word exists in. An example of such dependency relation is presented in Figure 6.
Example grams: $x_{i,d}[1]$ = `det`, $x_{i,d}[2]$ = `amod`, $x_{i,d}[3]$ = `nsubj`, …, $x_{i,d}[8]$ = `nmod`, where `det`, `amod`, `nsubj` and `nmod` are dependency relations.
- Application-specific patterns: input texts are considered as arrays of application-specific patterns (e.g., formatting codes). In this case, $x_{i,d}[g]$ denotes the g^{th} application-specific pattern in text $x_{i,d}$. The intuition is that texts that correspond to different labels are more likely to differ in their use of application-specific patterns than texts that correspond to the same labels.
Example text: suppose $x_{i,d}$ = "*[b][i]This[/i][/b] is a formatted text using [u]BB code[/u]*".
Example grams: $x_{i,d}[1]$ = "[b]", $x_{i,d}[2]$ = "[i]", $x_{i,d}[3]$ = "[/i]", $x_{i,d}[4]$ = "[/b]", …, $x_{i,d}[6]$ = "[/u]".
- Typos: input texts are considered as arrays of typos, where $x_{i,d}[g]$ denotes the g^{th} typo in text $x_{i,d}$.
Example text: suppose $x_{i,d}$ = "*Cna I see? teh byo siad. Yse. Of cuorse yuo cna*".
Example grams: $x_{i,d}[1]$ = "Cna", $x_{i,d}[2]$ = "teh", $x_{i,d}[3]$ = "byo", $x_{i,d}[4]$ = "siad", …, $x_{i,d}[8]$ = "cna".

[2]http://nlp.stanford.edu:8080/parser/index.jsp

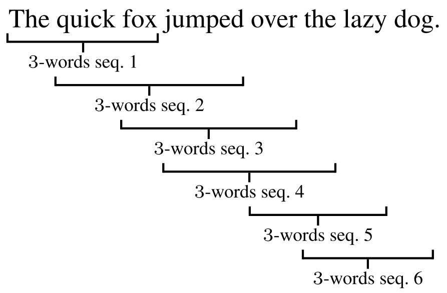

FIGURE 7. *n*-grams' sliding window for an example sentence where *n* = 3 and grams are *words*.

- Compound grams: theoretically, a gram could be defined as a tuple of multiple grams. Compound grams have the ability to capture the joint distribution of parts of texts taking specific gram values at the same time. For example, how many times was the word "*saw*" used as a noun? The following are examples of some compound grams that are made by two other grams:
  - Word-POS tag pairs: input texts are considered as arrays of word-POS tag pairs.
  Example grams: $x_{i,d}[1]$ = "The"-DT, $x_{i,d}[2]$ = "quick"-JJ, $x_{i,d}[3]$ = "fox"-NN, …, $x_{i,d}[8]$ = "dog"-NN.
  - Word-dependency relation pairs: input texts are considered as arrays of word-dependency relation pairs.
  Example grams: $x_{i,d}[1]$ = "The"-`det`, $x_{i,d}[2]$ = "quick"-`amod`, $x_{i,d}[3]$ = "fox"-`nsubj`, …, $x_{i,d}[8]$ = "dog"-`nmod`.

n: this parameter defines the width of the sliding window in the unit of grams. For example, if $n = 3$ and grams are *words*, then the sliding window width is three words as presented in Figure 7. It can be seen that the sliding window of a classical *n*-grams implementation moves *spatially* over the input texts.

len($x_{i,d}$): this parameter is the length of the text $x_{i,d}$ in grams. For example, if grams are words, then $\text{len}(x_{i,d}) = 8$ for the example text in Figure 7.

Then, all patterns (which, in this case, are sequences of *n* many grams; i.e., *n*-grams) can be found by any classical *n*-grams implementation. Let $ng_i[j]$ denote the j^{th} sequence of grams found by the searching algorithm from the input text $x_{i,d}$.

As an example, if $n = 3$ and grams are words, then all the found *n*-grams in the sentence "*the quick fox jumped over the lazy dog*" are: $ng_i[1]$ = (the, quick, fox), $ng_i[2]$ = (quick, fox, jumped), $ng_i[3]$ = (fox, jumped, over), $ng_i[4]$ = (jumped, over, the), $ng_i[5]$ = (over, the, lazy), and $ng_i[6]$ = (the, lazy, dog).

1) RELATION TO OUR NOTATION

For any text $x_{i,d}$, $\mathbf{x}_{i,d} = \text{fex}(x_{i,d})$ such that, for any $1 \leq j \leq \text{len}(x_{i,d}) - n + 1$, $\mathbf{x}_{i,d}[j]$ is a number that uniquely identifies $ng_i[j]$. To save space and reduce noisy features, *n*-grams that occur less than *l* many times can be discarded, where $l \in \mathbb{N}$.

Alternatively, $\mathbf{x}_{i,d}[j]$ can be defined as the frequency of $ng_i[j]$ in text $x_{i,d}$. To facilitate meaningful comparison between representations of different texts, the j^{th} component has to refer to the frequency of a specific n-gram consistently. This leads us to the problem of agreeing on the order of n-grams, which is usually addressed by agreeing on an arbitrarily-ordered list of n-grams as found in some reference corpus (e.g., the learning set $\mathcal{X}_L$). The order itself is not important; however, being consistent with the order is. Similar to the previous case, n-grams that occur less than l many times in reference texts can be discarded to preserve space and reduce noise.

2) DISCUSSION

Despite the simplicity of n-grams, their use often leads to the highest gains in classification accuracy relative to other data representation methods. Additionally, since finding n-grams is mostly language-independent (depending on how we define the grams), most n-grams implementations can be applied to any language.

On the other hand, n-grams assume that only patterns made of adjacent grams are helpful patterns. This is usually true (especially with natural languages); however, it is not always true. This is often a limitation that is found in n-grams as they can only locate those patterns that are made of adjacent n-grams. Hence, other data representations are proposed in the literature, some of which are variations of n-grams.

C. *k*-SKIP *n*-GRAMS

k-skip n-grams aim at generalizing classical n-grams such that grams within an n-gram need no longer be adjacent to each other in text $x_{i,d}$. This is accomplished by permitting up to k many skips between each pair of adjacent grams in an n-gram as presented in Algorithm 1, where $\mathbf{k}$ is a tuple with $n-1$ elements and $\mathbf{k}[j]$ denotes the j^{th} element of the tuple $\mathbf{k}$. For example, if $k = 2$, $n = 3$, and grams are words, then the found k-skip n-grams in the sentence "*the quick fox jumped over the lazy dog*" are: $kng_i[1] = (\text{the, quick, fox})$, $kng_i[2] = (\text{the, fox, jumped})$, $kng_i[3] = (\text{the, jumped, over})$, …, $kng_i[25] = (\text{over, the, lazy})$, $kng_i[26] = (\text{over, lazy, dog})$, $kng_i[27] = (\text{over, the, dog})$, and $kng_i[28] = (\text{the, lazy, dog})$.

Algorithm 1 Pattern Search by k-Skip n-Grams

```
for all $\mathbf{k} \in \{0, \dots, k\}^{n-1}$, such that $0 \leq \sum_{j=1}^{n-1} \mathbf{k}[j] \leq k$ and $n + \sum_{j=1}^{n-1} \mathbf{k}[j] \leq \text{len}(x_{i,d})$ do
    for all $j \in \{1, \dots, \text{len}(x_{i,d}) - \left(n + \sum_{j=1}^{n-1} \mathbf{k}[j]\right) + 1\}$ do
        $kng_i[j] = \left(x_{i,d}[j], x_{i,d}\left[j+1+\mathbf{k}[1]\right], x_{i,d}\left[j+2+\mathbf{k}[1]+\mathbf{k}[2]\right], \dots, x_{i,d}\left[j+n-1+\sum_{j=1}^{n-1} \mathbf{k}[j]\right]\right)$
    end for
end for
```

Note that the skips are only used to add an amount of tolerance (up to k skips) regarding the grams adjacency within an n-gram; that is, depending on the value of k, the grams in a sequence need no longer be necessarily adjacent to each other. However, such skips are not encoded in the identified n-grams.

1) RELATION TO OUR NOTATION

Since the skips are not encoded in k-skip n-grams, their structure is identical to that of the classical n-grams.

2) DISCUSSION

The advantage of k-skip n-grams is that they can identify patterns of grams that are not adjacent to each other (in addition to identifying those that are adjacent). In other words, k is a parameter that introduces a degree of tolerance by which patterns that are made of non-adjacent grams are identified.

The total number of patterns with exactly s skips that k-skip n-grams identify are:

$$\begin{cases} \text{len}(x_{i,d}) - n + 1 & \text{if } n = 1 \\ \binom{n-2+s}{n-2}\left(\text{len}(x_{i,d}) - (n+s) + 1\right) & \text{if } n > 1 \end{cases} \tag{14}$$

where $\binom{n-2+s}{n-2}$ is a binomial coefficient.

However, since k-skip n-grams identify all patterns with all $s \in \{0, 1, \dots, k\}$, the total number of patterns that k-skip n-grams find are:

$$\begin{cases} \text{len}(x_{i,d}) - n + 1 & \text{if } n = 1 \\ \sum_{s=0}^{k} \left(\binom{n-2+s}{n-2}\left(\text{len}(x_{i,d}) - (n+s) + 1\right)\right) & \text{if } n > 1 \end{cases} \tag{15}$$

Therefore, the disadvantage is that this degree of tolerance is limited by up to only k skips. Addressing this limitation by choosing larger k values can be computationally too demanding to be feasible. This is because the total number of identified gram sequences explodes combinatorically as a function of k, as shown in (14) and (15).

D. SYNTACTIC *n*-GRAMS

Classical n-grams identify gram sequences based on their order of appearance in their source texts. I.e., n-grams are made of spatially adjacent grams.

However, syntactic n-grams propose to read the grams based on the grams order in syntactic representations of their source texts. I.e., grams in syntactic n-grams no longer need to be spatially adjacent in their source texts but rather adjacent in the syntactic tree representation of their source texts.

For example, to identify syntactic n-grams from the text "*the quick fox jumped over the lazy dog*", we perform the following steps in order:

1) Represent the input sentence into a syntactically parsed tree. The most commonly suggested syntactic tree representation for syntactic n-grams is the *dependency-based parse trees* [29]. Figure 6 presents such a dependency-based parse tree for the example sentence.

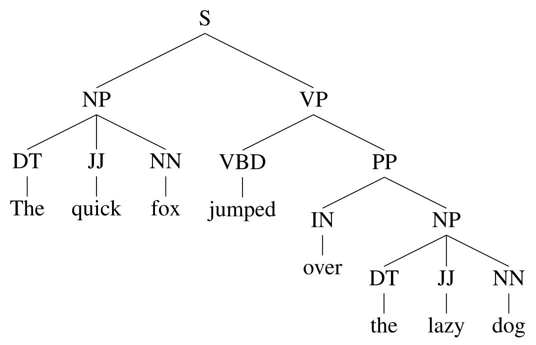

FIGURE 8. **Constituency-based parse tree for the sentence *"the quick fox jumped over the lazy dog"* as identified by Stanford's statistical sentence parser.**

2) Then, identify n-grams such that their grams are adjacent in the parsed tree. This can often lead to n-grams that contain grams that are not spatially adjacent.

For example, when $n = 2$ and grams are words, all found syntactic n-grams in the sentence above are found by recursively walking down the dependency tree in Figure 6 from its root as follows: $sng_i[1] =$ (jumped, fox), $sng_i[2] =$ (fox, quick), $sng_i[3] =$ (fox, the), $sng_i[4] =$ (jumped, dog), $sng_i[5] =$ (dog, lazy), $sng_i[6] =$ (dog, the), and $sng_i[7] =$ (dog, over).

Similarly, when $n = 3$, then all found syntactic n-grams for the dependency tree in Figure 6 are: $sng_i[1] =$ (jumped, fox, quick), $sng_i[2] =$ (jumped, fox, the), $sng_i[3] =$ (jumped, dog, over), $sng_i[4] =$ (jumped, dog, the), and $sng_i[5] =$ (jumped, dog, lazy).

Note that for this specific example dependency-based tree, no syntactic n-grams exist for $n > 3$.

Alternatively, one can substitute the dependency-based tree by a constituency-based tree [29] as presented in Figure 8. In this case when a constituency-based tree is constructed from the example sentence, examples of syntactic n-grams when $n = 3$ and grams are words are: $sng_i[1] =$ (S, NP, DT), $sng_i[2] =$ (NP, DT, The), $sng_i[3] =$ (NP, JJ, quick), $sng_i[4] =$ (NP, NN, fox), $sng_i[5] =$ (S, NP, JJ), $sng_i[6] =$ (S, NP, NN), $sng_i[7] =$ (S, VP, VBD), and $sng_i[8] =$ (VP, VBD, jumped).

1) RELATION TO OUR NOTATION

Similar to k-skip n-grams, syntactic n-grams do not encode the skips in their n-grams, they maintain the same vector space representation as both k-skip n-grams and classical n-grams.

2) DISCUSSION

The advantage of syntactic n-grams from the perspective of stylometry analysis is that they can identify patterns that are not necessarily spatially adjacent while keeping the number of identified patterns relatively small (i.e., avoids the combinatoric explosion of the number of patterns that k-skip n-grams face).

However, for stylometry analysis, their disadvantages are that, unlike classical and k-skip n-grams, syntactic n-grams might not identify some spatially adjacent gram sequences. This is because syntactic n-grams strictly walk over syntactic trees, resulting in missing some potentially important (for stylometry analysis) spatially-adjacent gram sequences. To address this, one may use syntactic n-grams to complement classical or k-skip n-grams.

Additionally, syntactic n-grams require sentence parsers, which are language-dependent. This can limit the applicability of syntactic n-grams to only languages with sentence parsers.

E. A GENERALIZATION OF n-GRAM-BASED METHODS

All n-gram-based feature extraction methods (i.e., classical, skip, and syntactic n-grams) can be modeled as special cases of at least l-frequent `dir`-directed k-skip n-grams, where:

- `dir` is the movement direction of the sliding window, as depicted in Figure 7. In classical n-grams, the direction is *spatial*, while in syntactic n-grams, the direction is either a *dependency-based tree* or a *constituency-based tree*. For brevity, we will refer to these directions as `spatial`, `deptree`, and `constree`, respectively.
- l specifies the minimum number of times a given sequence of grams must occur. For example, if $l = 5$, only those sequences that occur five times or greater will represent text samples.
- k, n, and grams are as defined in previous sections.

F. REWRITE RULES

From the perspective of generative grammars, texts can be generated by applying rewrite rules in a particular order. In the context of feature extraction in electronic text stylometry, the objective is to identify rewrite rules that could have generated input texts.

The use of rewrite rules in the literature of electronic text stylometry is restricted to Context-Free Grammars (CFGs) found by constituency-based parse trees.

For example, Figure 8 depicts a constituency-based tree of the sentence "*The quick fox jumped over the lazy dog*" from which the following example rewrite rules are found: S $\rightarrow$ NP + VP, NP $\rightarrow$ DT + JJ + NN, DT $\rightarrow$ The, JJ $\rightarrow$ quick, and NN $\rightarrow$ fox.

To reduce the amount of irrelevant information available by the terminal nodes (e.g., content words), rewrite rules that lead to terminal nodes could be removed.

Alternatively, it is possible to substitute such terminal nodes with other values to decide which information is kept and which is discarded. For example, if we represent the terminal nodes by their word shapes, then the rewrite rules for the example sentence will include S $\rightarrow$ NP + VP, NP $\rightarrow$ DT + JJ + NN, DT $\rightarrow$ Ccc, JJ $\rightarrow$ ccccc, and NN $\rightarrow$ ccc.

Note how terminal nodes "*The*" and "*quick*" are substituted by their corresponding shapes "*Ccc*" and "*ccccc*", respectively.

1) RELATION TO OUR NOTATION

For any text $x_{i,d}$, $\mathbf{x}_{i,d} = \text{fex}(x_{i,d})$ such that, for any j, $\mathbf{x}_{i,d}[j]$ is a number that uniquely identifies a specific rewrite rule that was used to generate some part of $x_{i,d}$. To save space and reduce noisy features, rewrite rules that occur less than l many times can be discarded, where $l \in \mathbb{N}$.

Alternatively, $\mathbf{x}_{i,d}[j]$ can be defined as the frequency of the rewrite rule uniquely identified by j.

Similar to previous features, to conveniently facilitate meaningful comparisons between representations of different input texts, the j^{th} component of any such vector representation should consistently refer to the frequency of the same rewrite rule.

2) DISCUSSION

The advantages of rewrite rules as features is that they can capture syntactic structures in text $x_{i,d}$, especially when considering discarding irrelevant information that exists in terminal nodes (by discarding them, or by substituting them by other values, such as word shapes, word lengths, and function words).

However, they share similar disadvantages with syntactic n-grams as they both rely on language-dependent sentence parsers.

G. RAW TEXT

While currently uncommon in the domain of stylometry problems, some stylometry-problem solvers expect as input raw texts to construct language models for the problem labels (e.g., authors).

Such stylometry methods are primarily inspired by the work of Tomàš Mikolov *et al.* on the construction of language models via Recurrent Neural Networks (RNNs) [30]. A notable example of a stylometry problem solver that analyzes raw text is the work of Douglas Bagnall [31], where raw texts were analyzed to construct language models, using RNNs, on a per-author basis to estimate the likelihood of each of such language models generating streams of letters that match the questioned test texts. To avoid constructing models that over-fit the training texts, specific information was removed from the input texts at a pre-processing stage (e.g., replacing all numbers with a placeholder).

1) RELATION TO OUR NOTATION

If $x_{i,d}$ is a raw text and $\mathbf{x}_{i,d}$ is a vector representing it (i.e., $\mathbf{x}_{i,d} = \text{fex}(x_{i,d})$), then, for any $j \in \{1, \ldots, \text{len}(x_{i,d})\}$ (recall that $\text{len}(x_{i,d})$ is the length of text $x_{i,d}$ in the unit of grams), conventional data representation methods define the j^{th} component $\mathbf{x}_{i,d}[j]$ to have a value that represents the frequency of a specific pattern j (e.g., some string) as measured across the input text $x_{i,d}$ as a whole.

However, the raw text representation method can be thought of as a special case of fex where the j^{th} component $\mathbf{x}_{i,d}[j]$ has a value that uniquely represents the j^{th} character in the text $x_{i,d}$, such that $\text{len}(x_{i,d})$ is the total number of characters in $x_{i,d}$.

For example, if $x_{i,d} =$ "*the quick fox jumped over the lazy dog*", then $\mathbf{x}_{i,d} = (116, 104, \ldots, 103)$ where each component represents a unique decimal value of the corresponding character in the input text $x_{i,d}$.

2) DISCUSSION

The raw text representation method can allow the classification algorithm to learn useful high-level features on its own, and learn a classification model based on such features. This can be advantageous as it will enable to identify high-level features that are counter-intuitive to humans but useful for the classification task at hand.

On the other hand, as computational time and space constraints exist in practice, such algorithms may miss some useful high-level features that are easily identified by the intuition of humans.

V. STYLOMETRY PROBLEMS AND SOLVERS

The problems can be defined as follows:

- Solvers of AA and AP problems are special cases of SCC if no domain variation exists between learning and testing samples and MCC if domains are allowed to vary among the learning and testing sets. The only distinction between AA and AP is that AA defines the targeted labels set $\mathcal{Y}_q$ as the set of author identities, while AP defines it as the set of author profile attributes. For example, in the case of age-group detection, $\mathcal{Y}_q = \{10\text{s}, 20\text{s}, \ldots\}$.
- Solvers of AV problems [32] are special cases of SOC if no domain variation exists among the analyzed samples, and MOC if otherwise. The targeted label set $\mathcal{Y}_q$ is defined as the set of author identities.
- Solvers of AC and AD problems are special cases of SOC if no domain variation exists among learning and testing samples, and MOC if domains are allowed to vary among the learning and testing sets. This is because such problems can be decomposed into multiple binary SOC and MOC problems. Both AC and AD define the targeted labels set $\mathcal{Y}_q$ as the set of author identities. The only distinction between AC and AD is that AC clusters text files, while AD clusters text parts (e.g., paragraphs).

Since the stylometry methods are not necessarily limited to specific data representation methods, the subsections below abstract the data representation stage by using the function fex as defined in Section III. I.e., for any text in domain d, $x_{i,d}$, its represented form is referred to by $\mathbf{x}_{i,d} = \text{fex}(x_{i,d})$, without specifying the specific representation.

The following points describe what is generally performed when solving stylometry problems:

- Use the stylometry problem-solver whose assumptions are most in line with the current problem's assumptions. For example, if a stylometry problem assumes a closed-set (e.g., SCC, MCC), then using a solver for closed-set scenarios will be more effective than a solver for open-sets (e.g., SOC, MOC). Open-set problem solvers can solve closed-set problems. However, they almost always tend to be less accurate than closed-set problem solvers because they ignore that the correct label is confined within the label's set of the training set. On the other hand, applying closed-set problem solvers on open-set stylometry problems will guarantee an incorrect answer if the correct label is not in the training set.

- Pre-process the involved texts to reduce the classification error of the stylometry problem solvers at later stages. For example, suppose it is known that the actual quantity of numbers in texts does not convey information about texts' authors. In that case, numbers could be deleted or replaced with a placeholder. Fundamentally, this pre-processing, where parts of texts are deleted or replaced by placeholders, could be considered a form of domain adaptation by data selection [33].
- If the stylometry problem solver requires texts to be represented as vectors, feature extraction methods, as surveyed in this paper, could be used to represent the texts as vectors, such that each vector's component is a feature's value for the text that the vector represents. Such text vectorization is also applicable to deep neural networks that process raw texts. The feature extraction methods can be supplied as input to the models alongside the pre-processed raw texts.
- If the involved texts fall under distinct domains, then domain adaptation methods could minimize the domain variation effect. Fundamentally, domain adaptation methods could be applied at every stage, such as pre-processing texts, text vectorization (or feature extraction), and the training phase of stylometry problem solver models.

A. GENERAL-PURPOSE LEARNING ALGORITHMS

Support Vector Machines (SVMs) are generally regarded among the most accurate general-purpose learning algorithms for solving SCC problems [18]. Examples of the use of such models in the literature are [34]–[37].

Decision trees are commonly used to solve SCC problems [38]–[40]. However, recently, decision trees were also used to solve AV problems (a special case of SOC problems) [41].

Similarly, Artificial Neural Networks (ANNs) are used to solve SCC problems [38], [42]–[45]. An interesting aspect of some ANN-based models is their ability to define the feature extraction function, fex, during the learning process. In other words, the learning algorithm optimizes the decision boundaries and optimizes the definition of its fex implementation. Consequently, the need for a manually-crafted fex implementation is removed. Instead, raw texts are fed into the model.

B. COMMON *n*-GRAMS

Keselj *et al.* proposed an estimation of SCC in [46] as follows:

1) Texts are represented by the frequency of some chosen classical n-grams. The chosen n-grams are those that at least occur in a single text sample for L many times. In other words, using our notation, for any $\mathbf{x}_{i,d} \in \mathcal{X}$, $\mathbf{x}_{i,d}$ is a multi-dimensional real vector such that $\mathbf{x}_{i,d}[j]$ represents the frequency of the j^{th} chosen n-gram. If for some $\mathbf{x}_{i,d} \in \mathcal{X}$ the j^{th} n-gram does not exist, then $\mathbf{x}_{i,d}[j] = 0$ is assumed.
2) A distance function $\text{cng} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ between two samples $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}$ is proposed as follows:

$$\text{cng}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \sum_{c \in \{1,2,\ldots\}} \left(\frac{2(\mathbf{x}_{i,d}[c] - \mathbf{x}_{j,d}[c])}{\mathbf{x}_{i,d}[c] + \mathbf{x}_{j,d}[c]} \right)^2$$

3) Then, for any disputed text $\mathbf{x}_i \in \mathcal{X}_T$, the SCC estimation of its true target classification label $y_{i,q}$, namely $\hat{y}_{i,q}$, is found as follows:

$$\hat{y}_{i,q} = y_{j,q}$$

where $y_{j,q}$ is the true target classification label of the text sample $\mathbf{x}_{j,d}$ that is found as follows:

$$\mathbf{x}_{j,d} = \arg\min_{\mathbf{x}_{i,d} \in \mathcal{X}_L} \text{cng}(\mathbf{x}_a, \mathbf{x}_{i,d})$$

In other words, the author of the testing text sample $\mathbf{x}_{i,d}$ is assumed to be the same author of the learning text sample $\mathbf{x}_{j,d}$ that achieves the lowest $\text{cng}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})$ value against the testing text sample $\mathbf{x}_{i,d}$ relative to the other texts in $\mathcal{X}_L$.

Common n-grams (CNG) advantage is their independence from the language of learning and testing text samples. This makes CNG also applicable for programming languages. A significantly simplified CNG variant is proposed in [47] by which the value $\text{cng}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})$ is simply substituted by the quantity of the L-most frequent common n-grams between inputs $\mathbf{x}_{i,d}$ and, $\mathbf{x}_{j,d}$.

C. COMPRESSION

For any samples $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}$ Keogh *et al.* propose in [48] the following similarity measurement function:

$$\text{cdm}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \frac{\circ(\mathbf{x}_{i,d} || \mathbf{x}_{j,d})}{\circ(\mathbf{x}_{i,d}) + \circ(\mathbf{x}_{j,d})}$$

where $\circ : \mathcal{X} \to [0, \infty)$ is a function that returns total number of bits after compressing its input using some compression algorithm (e.g., GZIP, and RAR) and $\mathbf{x}_{i,d} || \mathbf{x}_{j,d}$ is the concatenation of texts $\mathbf{x}_{i,d}$ and $\mathbf{x}_{j,d}$.

Another compression-based similarity measurement function is proposed by Cilibrasi *et al.* in [49] as follows:

$$\text{ncd}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \frac{\circ(\mathbf{x}_{i,d} || \mathbf{x}_{j,d}) - \min(\circ(\mathbf{x}_{i,d}), \circ(\mathbf{x}_{j,d}))}{\max(\circ(\mathbf{x}_{i,d}), \circ(\mathbf{x}_{j,d}))}$$

Similar to common n-grams, for any testing sample $\mathbf{x}_{i,d} \in \mathcal{X}_T$, such functions can be used to solve SCC problems by finding the estimation $\hat{y}_{i,q}$ as follows:

$$\hat{y}_{i,q} = y_{j,q}$$

where $y_{j,q}$ is the true target classification label of sample $\mathbf{x}_{j,d}$ that is found as follows:

$$\mathbf{x}_{j,d} = \arg\min_{\mathbf{x}_{i,d} \in \mathcal{X}_L} \mathrm{cdm}(\mathbf{x}_{i,d}, \mathbf{x}_{i,d})$$

or by substituting the cdm function by ncd.

D. BURROWS DELTA

Burrows delta [50] and its variants are among the most successful stylometry distance measures (often used to find estimations of solvers of SCC problems). Fundamentally Burrows delta is the distance function $\Delta_B : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ such that the distance is smaller when the input texts are more similar to one another.

Additionally, let $\mathrm{zscore}(\mathbf{x}_{i,d}[c]) = \frac{\mathbf{x}_{i,d}[c]-\mu_c}{\sigma_c}$ be the z-score of frequency $\mathbf{x}_{i,d}[c]$, μ_c be the frequency mean of feature (or component) c, and σ_c be the standard deviation of feature c.

Then, for any two represented texts $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}$, Burrows delta is defined by the Manhattan distance as follows:

$$\Delta_B(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \sum_{c \in \{1,2,\ldots\}} \left| \mathrm{zscore}(\mathbf{x}_{i,d}[c]) - \mathrm{zscore}(\mathbf{x}_{j,d}[c]) \right|$$

Variations of Burrows delta essentially substitute the Manhattan distance with other distance measures.

E. UNMASKING

Koppel and Seidman propose the unmasking algorithm as an estimated solver to the AV problem [32]. For any $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}_T$, the unmasking algorithm aims to answer whether $y_{i,q} = y_{j,q}$ (recall that $y_{i,q}$ is the classification label of the represented text $\mathbf{x}_{i,d}$, under classification task q).

Intuitively, the unmasking algorithm assumes that texts written by the same authors are harder to separate (or classify as different authors) than texts written by different authors. More specifically, the unmasking algorithm solves the AV problem as follows:

1) As stated in Section V, the AV problem assumes that, for any $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}_T$, $\mathbf{x}_{i,d}$, and $\mathbf{x}_{j,d}$ are collections of texts such that texts within each collection are written by the same author. If there is only one text in each text collection $\mathbf{x}_{i,d}$, and $\mathbf{x}_{j,d}$, then the unmasking method creates a collection of multiple texts by splitting each text into multiple parts. Therefore, input texts should be large enough to allow for the text parts to be large enough for subsequent analysis [37].
2) Text parts in $\mathbf{x}_{i,d}$ and $\mathbf{x}_{j,d}$ are assumed to correspond to target classification labels $y_{i,q}$ and $y_{j,q}$, respectively, such that $y_{i,q} \neq y_{j,q}$.
3) Using ten-fold cross-validation, SVM models are trained and tested to predict the class labels of the text parts in $\mathbf{x}_{i,d}$ and $\mathbf{x}_{j,d}$ based on their assumed clusters $y_{i,q}$ and $y_{j,q}$, respectively. Such ten-fold cross-validation is repeated multiple times, such as, at each attempt, a given number of the strongest features are removed. This results in degrading the classification accuracy as the strongest features are removed. When the accuracy of such ten-fold cross-validation evaluations is plotted as a function of each feature removal step, a classification *degradation curve* is found.
4) If the degradation curve is sufficiently steep, it is assumed that $y_{i,q} = y_{j,q}$, otherwise $y_{i,q} \neq y_{j,q}$ is assumed.

F. IMPOSTORS

Koppel *et al.* [51] propose the *impostors* algorithm to estimate a solver of the SOC problem. Intuitively, the impostors algorithm is an ensemble of randomized text similarity measurement functions that assume that input texts are written by the same authors if their similarity towards themselves is higher than their similarity towards other texts of other authors.

More specifically, for any test samples $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}_T$, the impostors algorithm aims to answer whether $y_{i,q} = y_{j,q}$ by the following steps:

1) A score is initialized: $s \leftarrow 0$.
2) A random subset of texts that fall under the domains of $\mathbf{x}_{i,d}$, and $\mathbf{x}_{j,d}$ are obtained. We refer to this collection of texts as the *in-domain texts*. With relation to our notation, this random subset can be perceived as a samples subset of the learning set $\mathcal{X}_L$ whose target classification labels are different than $y_{i,q}$, and $y_{j,q}$ (but their irrelevant classification task labels match as they are in the same domains as the testing samples).
3) Let $\mathrm{sim} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be some similarity distance function that returns a larger score the more similar its texts are, and a smaller score the less similar its input texts are. Then, the impostors algorithm finds the most similar in-domain text to $\mathbf{x}_{i,d}$, and $\mathbf{x}_{j,d}$. Let $\mathbf{m}_{i,d}$, and $\mathbf{m}_{j,d}$ be the most similar in-domain texts to $\mathbf{x}_{i,d}$, and $\mathbf{x}_{j,d}$, respectively.
4) The score is then updated as follows:

$$s \leftarrow s + \begin{cases} \frac{1}{r} & \text{if } \left(\begin{array}{c} \mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})^2 > \\ \mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{m}_{i,d}) \times \mathrm{sim}(\mathbf{x}_{j,d}, \mathbf{m}_{j,d}) \end{array} \right) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where $r \geq 1$. Note that the function sim compares the input texts from the perspective of a random subset of vector components (or features).

Other similar score aggregation methods have also been successfully evaluated in the literature, such as the following as adopted by Khonji and Iraqi [52]:

$$s \leftarrow s + \frac{\mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})^2}{\mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{m}_{i,d}) \times \mathrm{sim}(\mathbf{x}_{j,d}, \mathbf{m}_{j,d})} \tag{17}$$

5) Steps 2, 3, and 4 are repeated r times. If score s is greater than some threshold (which is to be found in the training phase), then $y_{i,q} = y_{j,q}$ is assumed; otherwise, $y_{i,q} \neq y_{j,q}$ is assumed.

The most notable contribution of this method is its attempt at normalizing the score of its similarity measure (i.e., sim function) based on in-domain texts, beyond simply analyzing the input questioned texts, which proved itself to be highly effective at solving AV problems, even when the similarity measure, sim, is defined to be as simple as the min-max distance measure (which cannot weigh the importance of features individually).

VI. KEY DRAWBACKS OF EXISTING SOLVERS

The following are common drawbacks for all stylometry problem solvers:

- Lack of knowledge about what the solver is truly solving. A solver is trained to solve, say, an author attribution problem. However, the solver has no property that makes it an author attribution model instead of a topic classification model. Usually, stylometry problem solvers are, in part, topic classification models. The reason is that the distribution of writing styles is not entirely independent of the topic. Currently, there are two typical methods to minimize this drawback: the use of feature sets with low topic information, such as the distribution of function words, and the use of topic-controlled datasets. However, none of the methods are adequate, as the former often also results in degraded classification accuracy, and the latter is often too difficult to obtain in reality.
- Vulnerability against adversarial attacks. Style obfuscation and imitation performed by untrained individuals can degrade the classification accuracy of stylometry problem-solvers down to random chance guessing and below random chance guessing, respectively [53].
- Degraded accuracy against cross-domain stylometry problems. The classification accuracy of the solvers degrades significantly when given texts that fall under a domain other than the domain of texts that they were trained on. This is specifically a challenge for open-set problem scenarios (i.e., MOC) due to insufficient domain adaptation techniques.

The following are solver-specific drawbacks:

- Extrinsic AV problem solvers, such as the Impostors method, require external datasets that represent texts that fall in the same domain as those of the questioned texts. Obtaining in-domain texts is an open research problem at the moment. Currently, used methods are mainly rough estimations of the in-domain samples and can be expensive to obtain. For example, an approach that is commonly used with the Impostors method is to query a search engine over the Internet to get web pages with related texts, followed by using heuristics to extract the text from the web page. The result is often a very noisy text set, possibly with code snippets. One of the main factors that cause the Impostors method to be among the slowest AV solvers is that it analyzes such noisy in-domain samples multiple times (depending on the chosen parameters).
- Simple distance measures, such as common n-grams, compression-based distances, and Burrows delta, are among the fastest in terms of their run-time speed. However, while the cost-accuracy trade-off of such measures can be attractive, they are no longer among the best performing in terms of their classification accuracy.
- AV problem solvers based on neural network language models are among the computationally demanding solvers in this domain. For every unknown text, the neural network is trained to learn a language model for the text.

VII. FEATURES EVALUATION METHODOLOGY

In Section IV-E, we generalized many feature extraction methods (i.e., special cases of the fex function) as special cases of the at least l-frequent `dir`-directed k-skip n-grams. This generalization simplified our implementation of the many features while simultaneously allowing the identification of previously unevaluated features.

One of the critical gaps in the literature's current state is the lack of joint evaluation of the many feature extraction methods. While the features are evaluated in isolation, it is unknown how they compare against the other features. This is because the evaluations follow non-unified testing beds. For example, they may be using different testing datasets and different evaluation methodologies.

Another issue is that most of the existing evaluations derive conclusions without performing statistical significance tests. Therefore, even within their isolated testing beds, it is unknown whether their outcomes are due to a systematic difference in the evaluated methods as opposed to sampling noise due to random chance.

Additionally, many of the developed methods are often inaccessible to the community. This results in slowing down the pace of research as different research groups would need to re-implement the methods.

Our goal in this evaluation is to:

- Evaluate the many feature extraction methods under a unified testing bed using multiple datasets. This way, we can compare their performance jointly.
- Perform statistical significance tests to objectively identify the probability of having observed evaluation outcomes arise under the null hypothesis (i.e., the p value). This is necessary to derive any conclusions from the evaluation results.
- Permit the reproducibility of this evaluation and the reusability of our developed tools by releasing all our associated evaluated datasets and tools openly under a permissible open source license. Additionally, our tools are implemented by a friendly programming language (e.g., Python).

A. *EVALUATED* fex *IMPLEMENTATIONS*

This evaluation implements many fex implementations as a special case of at least l-frequent `dir`-directed k-skipped n-grams by exhaustively varying its parameters as follows:

- All $l \in \{1, 2, \ldots, 99\}$.
- All $\texttt{dir} \in \{\texttt{spatial}, \texttt{deptree}\}$.

TABLE 2. Tested variables.

Variable	Description	Used values
l	The minimum number of times a given sequence of grams must occur.	$1, 2, \dots, 99$
dir	The movement direction of the sliding window.	`spatial`, `deptree`
k	The maximum allowed number of skips between each pair of adjacent grams.	$0, 1, 2, 3$
n	The width of the sliding window in the unit of grams.	$1, 2, 3$
gram	The unit of the processed text.	`dep`, `funcword`, `pos`, `wordlen`, `wordshape`, `word`, `pos-dep`, `word-dep`, `word-pos`, `wordshape-word`

- All $k \in \{0, 1, 2, 3\}$.
- All $n \in \{1, 2, 3\}$.
- All gram $\in$ { `dep`, `funcword`, `pos-dep`, `pos`, `word-dep`, `word-pos`, `wordlen`, `wordshape-word`, `wordshape`, `word`}.

This resulted in a total number of $99 \times 2 \times 4 \times 3 \times 10 = 23,760$ unique implementations of the features extraction function fex. Table 2 lists the tested variable and their values.

B. EVALUATION PROBLEMS

The evaluation problems follow the SCC scenario. Recall that SCC stylometry problems are those where the target labels of testing samples are guaranteed to exist in the set of target labels of the learning samples while simultaneously assuming that learning and testing samples are drawn from the same domain.

We used the following text datasets to create SCC problems:

- S24: Problem C of the PAN author identification competition. This is an author identification scenario, and the set of labels are author identifiers. Since this dataset is composed of 24 text files, we refer to it by S24.
- S1000: This is a reduced version of the one used in [54]. This dataset is composed of a set of IMDb reviews as authored by some of the "prolific" users of IMDb [54]. Initially, this dataset is composed of 62,000 text files in total. However, we reduced the dataset down to 1,000 files by eight authors due to the exhaustive nature of our evaluation and its associated computational constraints. The selection of this reduced subset was uniformly random. The reason for this reduction is because our assessment of the parameters of the feature extraction functions is exhaustive, and performing this with larger datasets was not computationally feasible.

Various statistics of the datasets S24 and S1000 are presented in Table 3.

For each dataset, the SCC problems are constructed as follows:

- Two third of the text files are used as the learning set. This learning set, along with the corresponding target classification label of each file (i.e., the author identifier of each file), is fed into a Random Forest (RF) classification learning algorithm. The output of this RF learning algorithm is an SCC classification model.
- The remaining one-third of the text files are used as testing samples. Target classification labels of these testing samples are predicted by the RF model that is trained earlier. The prediction output of this RF model is logged for future analysis as detailed in Section VII-C.

TABLE 3. Datasets statistics.

Dataset name	S24	S1000
Dataset source	PAN'12 Prob. C [55]	IMDb [54]
#Text files	24	1,000
#Authors	8	8
#Text files per author	3	125
Avg. #words per file	5,361.75	399.366
Avg. #letters per file	30,115.9583	2,058.718

To more efficiently utilize the datasets S24 and S1000, we repeat the steps above by using 3-fold cross-validation. This enables us to effectively test against all the text files in a given dataset while simultaneously ensuring that the learned RF models (in each fold) are never trained by any text in the testing fold.

Note that due to the size of S1000, it became computationally infeasible for us to evaluate fex implementations for when $l = 1$ due to the sheer amount of identified patterns. Therefore, for S1000, we evaluate implementations of fex for all $l \in \{2, 3, \dots, 99\}$. Note how $l \neq 1$ implies that the patterns that occur only once will be discarded (only those that occur more than once will be considered). This ensures that we will not identify too many patterns that cannot fit in memory. This also means that we evaluate $98 \times 2 \times 4 \times 3 \times 10 = 23,520$ fex implementations on the S1000 dataset. This limitation does not affect S24 as the dataset is considerably smaller. Therefore we evaluate all of the 23,760 distinct fex implementations on S24.

C. EVALUATION METRICS

Let fex$_i$ and fex$_j$ be any two distinct implementations of feature extraction functions. We represent the evaluation datasets once by using fex$_i$ and another by using fex$_j$. This gives us two different representations of the datasets.

Additionally, let $\texttt{RF}_i$ and $\texttt{RF}_j$ be two distinct RF classification models. By feeding the testing samples to the classification models $\texttt{RF}_i$ and $\texttt{RF}_j$, we obtain their prediction outputs O_i and O_j, respectively.

To evaluate the effectiveness of the feature extraction functions fex$_i$ and fex$_j$, we measure the following:

- The accuracy of their classification models $\texttt{RF}_i$ and $\texttt{RF}_j$, respectively. More specifically, for any i, the accuracy of O_i is measured as follows:

$$\underset{i}{\text{acc}} = \frac{\sum_{o \in O_i} \begin{cases} 1 & \text{if prediction } o \text{ is correct} \\ 0 & \text{otherwise} \end{cases}}{\text{total number of problems}} \quad (18)$$

TABLE 4. Statistical significance levels.

Symbol	Level	Name
=	$p \geq 0.05$	Significance not shown
*	$0.05 > p \geq 0.01$	Statistically significant
**	$0.01 > p \geq 0.001$	Very statistically significant
***	$p < 0.001$	Highly statistically significant

- The statistical significance of the difference in the measured accuracy of the models $\texttt{RF}_i$ and $\texttt{RF}_j$, namely acc_i and acc_j, respectively. This is to identify whether the observed differences are statistically significant. Specifically, we define the following hypothesis:
  - H_0: the differences between acc_i and acc_j are due to random noise. This is also referred to as the *null hypothesis*.
  - H_1: the differences between acc_i and acc_j are because of a significant difference in the feature extraction methods fex_i and fex_j. Since the classification models $\texttt{RF}_i$ and $\texttt{RF}_j$ differ only by their implementation of the feature extraction functions, any systematic difference has to be because of the selection of the feature extraction functions. This is also referred to as the *alternative hypothesis*.

  Then, we measure the probability that the observed absolute difference of the accuracy, namely $|\,\text{acc}_i - \text{acc}_j\,|$, or greater absolute differences, can arise under hypothesis H_0. We refer to this probability as the p value.

To measure the p value, we have to identify the distribution of absolute accuracy differences under the null hypothesis H_0. We adopt the statistical significance naming convention from [24] as presented in Table 4.

VIII. FEATURES EVALUATION RESULTS

This evaluation aims to identify properties of the feature extraction functions that correspond to the increase in classification accuracy. Since this evaluation tests many feature extraction functions that are special cases of the at least l-frequent $\texttt{dir}$-directed k-skipped n-grams, the properties that we evaluate their effects on the classification accuracy are l, $\texttt{dir}$, k, n, and grams.

Additionally, since the at least l-frequent $\texttt{dir}$-directed k-skipped n-grams is a generalization of the following previously known feature extraction methods:

1) Distribution of grams.
2) Distribution of n-grams: This is a generalization of grams by which the frequency of n sequences of adjacent grams are measured.
3) Distribution of k-skipped n-grams: This is a generalization of n-grams where skips up to k are tolerated. Therefore n sequences of grams need no longer be adjacent and can have up to k skips between them.
4) Distribution of syntactic n-grams: This is a generalization over n-grams where we are no longer limited to scan a text for gram sequences spatially but rather scan a text by following a syntactic path.

We also take this opportunity to attempt to answer the following questions: do the various generalizations above benefit the accuracy of stylometry problem solvers? Do they enable the stylometry problem solvers to identify more accurate classification models? If yes, then which of the generalizations benefit the accuracy of stylometry solvers?

We find answering these questions of importance as those generalizations are used in the literature of stylometry problems while never being jointly evaluated yet.

A. INDEPENDENT PARAMETERS EVALUATION

As discussed in Sections VII-A and VII-B, a total number of $23,760$ distinct feature extraction functions are implemented and used to represent texts of the evaluation datasets. This process results in $23,760$ distinct representations of the evaluation datasets. Then, for each of the distinct representations of the evaluation datasets, the RFs algorithm is used to construct AA classification models, and their classification accuracy is measured.

The process above results in $23,760$ classification accuracy measurements, each of which represents the accuracy that was achieved when using a specific feature extraction function to represent the evaluation datasets. Since the number of parameters that define the feature extraction functions is 5, one would need 6 dimensions to represent the complete results in a single figure. However, due to significant challenges in representing items in 4, or higher dimensional spaces, this subsection will present the accuracy measurements independently for each parameter. A joint analysis will be presented in later sections.

Specifically, each figure in this section represents empirical commutative density functions (ECDFs) of the classification accuracy measurements, such that each ECDF corresponds to a specific value of a specific parameter that defines feature extraction functions. In other words, the horizontal axis represents the classification accuracy measurement values, and the vertical axis represents commutative probability values. For example, suppose the parameter is the definition of a gram. In that case, an ECDF curve is presented for when grams are defined to be words, another ECDF curve is presented for when grams are POS tags, and so on with the rest of the evaluated gram definitions.

Such ECDFs can be used to observe the distribution of classification accuracy measurements from the perspective of various values of a specific feature extraction function parameter. For example, one could identify which parameter values allow for the existence of the highest classification accuracy values.

1) PARAMETER l

Figure 9 presents the classification accuracy of each of the $23,760$ RF classification models on dataset S24 from the perspective of parameter l. The figure shows 99 curves, each of which is the ECDF of the many classification accuracies of all of the RF classification models that share the same value of the parameter l. In other words, each of the ECDFs represents

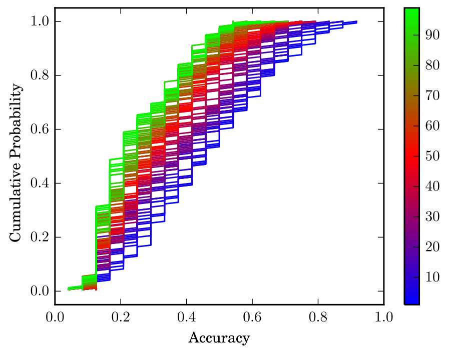

FIGURE 9. The ECDFs of classification accuracy from the perspective of the parameter l (for all $l \in \{1, 2, \ldots, 99\}$ against the S24 problems set).

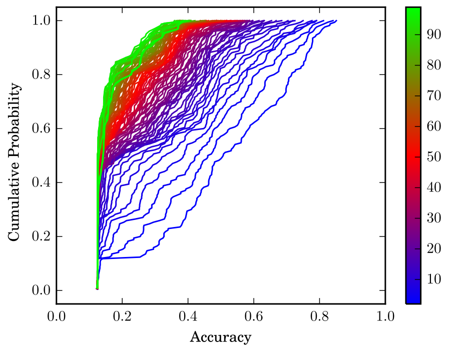

FIGURE 10. The ECDFs of classification accuracy from the perspective of the parameter l (for all $l \in \{1, 2, \ldots, 99\}$ against the S1000 problems set).

the accuracy of $23,760/99 = 240$ RF many classification models.

Similar to Figure 9, Figure 10 represents the same, except for evaluations on the dataset S1000. Note that the ECDFs in Figure 10 are considerably smoother than those in Figure 9. This is because the dataset S1000 is composed of 1,000 SCC author identification problems (which means that the classification accuracy measurements take values in $\{\frac{0}{1{,}000}, \frac{1}{1{,}000}, \ldots, \frac{1{,}000}{1{,}000}\}$), whereas the dataset S24 is composed of only 24 of such problems (which means that the classification accuracy measurements take values in $\{\frac{0}{24}, \frac{1}{24}, \ldots, \frac{24}{24}\}$, which is only 24 possible values and therefore the approximately 24 stair steps).

We can see from the ECDFs in Figures 9 and 10 that the evaluations on both of the datasets, S24 and S1000, agree that the selection of lower values of l can allow for the identification of more accurate classification models.

Specifically, Tables 5 and 6 list the highest classification accuracy for each $l \in \{1, 2, \ldots, 4\}$, along with their corresponding pair-wise p values (note that $l = 1$ is not evaluated on the dataset S1000 for the reason given in Section VII-B).

It can be seen from Tables 5 and 6 that both of the datasets agree in that lower values of l can allow for higher classification accuracy levels. However, due to the small size of S24, none of the differences in the accuracy levels in Table 5 are statistically significant ($p \geq 0.05$). However, thanks to the larger size of the dataset S1000, Table 6 can show that the increase in the classification accuracy with $l \in \{2, 3\}$, relative to $l = 4$, is statistically highly significant ($p < 0.001$).

TABLE 5. p values of the most accurate methods from the perspective of the parameter l. To save space, only $l \in \{1, 2, 3, 4\}$ is shown (S24 problems set).

l	2 acc $= 0.92$	3 acc $= 0.88$	4 acc $= 0.88$
1 acc $= 0.92$	$p = 1.0000$ =	$p = 0.5295$ =	$p = 0.5005$ =
2 acc $= 0.92$	–	$p = 0.5045$ =	$p = 0.5155$ =
3 acc $= 0.88$	–	–	$p = 1.0000$ =

TABLE 6. p values of the most accurate methods from the perspective of the parameter l. To save space, only $l \in \{2, 3, 4\}$ is shown (S1000 problems set).

l	3 acc $= 0.84$	4 acc $= 0.81$
2 acc $= 0.85$	$p = 0.1818$ =	$p = 0.0010$ ***
3 acc $= 0.84$	–	$p = 0.0020$ **

However, it is essential to note that the parameter l is, fundamentally, a features selection parameter. I.e., larger values of l will cause more features to be eliminated, and smaller values of l will present more features to the learning algorithm. The fundamental question here is whether the learning algorithm can identify and use robust features while ignoring harmful (or useless) features.

2) PARAMETER `dir`

When the parameter $n = 1$, the `dir` parameter is irrelevant. This is because `dir` determines the direction by which *multiple* grams are identified to form a single n-gram. For such cases when `dir` is irrelevant, we denote them by "N/A".

Figures 11 and 12 presents the ECDFs of the classification accuracy levels from the perspective of values of the parameter `dir` as evaluated on datasets S24 and S1000, respectively.

It can be seen that both Figures 11 and 12, on datasets S24 and S1000, agree that higher classification accuracy levels can be achieved when `dir` = `spatial` than when `dir` = `deptree`. Additionally, Tables 7 and 8 show that such differences are statistically significant in dataset S24 ($p = 0.023$) and statistically highly significant in dataset S1000 ($p < 0.001$).

It can also be seen that Figures 11 and 12 disagree on the effectiveness of `dir` = N/A. Figure 11 suggests that `dir` = `spatial` can allow for the identification classifiers that are more accurate than those identified by the case when `dir` = N/A, while Figure 12 indicates the opposite. However, Table 7 shows that the observation from Figure 11 is not statistically significant, while Table 8 shows that the observation from Figure 12 is statistically significant.

Additionally, since the parameter `dir` is one of the parameters that are responsible for generalizing n-grams, k-skip n-grams, and syntactic n-grams with dependency trees, we can extend the findings for the parameter `dir` to answer some of Section VIII's questions.

We know that n-grams and k-skip n-grams follow the `spatial` direction, while syntactic n-grams with

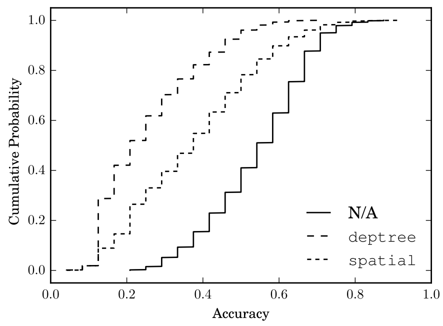

FIGURE 11. The ECDFs of classification accuracy from the perspective of the sliding window movement direction of *n*-grams (S24 problems set).

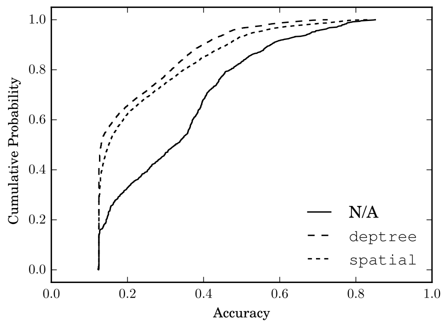

FIGURE 12. The ECDFs of classification accuracy from the perspective of the sliding window movement direction of *n*-grams (S1000 problems set).

TABLE 7. *p* values of the most accurate methods from the perspective of the sliding window movement direction of *n*-grams (S24 problems set).

dir	deptree acc $= 0.71$	spatial acc $= 0.92$
N/A acc $= 0.88$	$p = 0.1528$ $=$	$p = 0.7193$ $=$
deptree acc $= 0.71$	–	$p = 0.0230$ *

TABLE 8. *p* values of the most accurate methods from the perspective of the sliding window movement direction of *n*-grams (S1000 problems set).

dir	deptree acc $= 0.73$	spatial acc $= 0.83$
N/A acc $= 0.85$	$p = 0.0009$ ***	$p = 0.0609$ $=$
deptree acc $= 0.73$	–	$p = 0.0009$ ***

dependency trees follow the `deptree` direction. Therefore, having both of the datasets, S24 and S1000, agree that more accurate classification models can be identified with `dir` $=$ `spatial` than with `dir` $=$ `deptree` is an indication that n-grams or k-skip n-grams are superior to syntactic n-grams with dependency trees concerning their ability in identifying features that result in higher classification accuracy.

3) PARAMETER *k*

While Figures 13 and 14 do not show a clear pattern, it can be seen that lower values of k allow for achieving equal or higher degrees of classification accuracy than the case when k is larger. Specifically, maximum accuracy is achieved in dataset S24 when $k \in \{0, 1\}$, and the same achieved in dataset S1000 when $k = 0$. However, as shown in Tables 9 and 10,

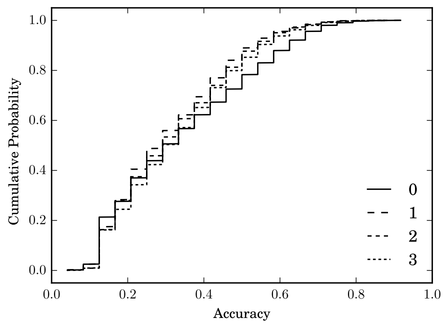

FIGURE 13. The ECDFs of classification accuracy from the perspective of the parameter *k* (S24 problems set).

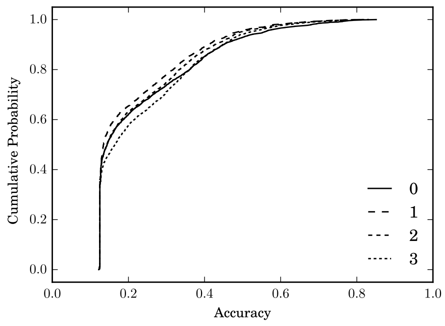

FIGURE 14. The ECDFs of classification accuracy from the perspective of the parameter *k* (S1000 problems set).

TABLE 9. *p* values of the most accurate methods from the perspective of the parameter *k* (S24 problems set).

k	1 acc $= 0.92$	2 acc $= 0.88$	3 acc $= 0.88$
0 acc $= 0.92$	$p = 1.0000$ $=$	$p = 0.5035$ $=$	$p = 0.4965$ $=$
1 acc $= 0.92$	–	$p = 0.4975$ $=$	$p = 0.4765$ $=$
2 acc $= 0.88$	–	–	$p = 1.0000$ $=$

TABLE 10. *p* values of the most accurate methods from the perspective of the parameter *k* (S1000 problems set).

k	1 acc $= 0.82$	2 acc $= 0.83$	3 acc $= 0.83$
0 acc $= 0.85$	$p = 0.0050$ **	$p = 0.0500$ *	$p = 0.0529$ $=$
1 acc $= 0.82$	–	$p = 0.0460$ *	$p = 0.0549$ $=$
2 acc $= 0.83$	–	–	$p = 0.8961$ $=$

none of this is statistically significant in dataset S24, while the superiority of $k = 0$ over $k > 0$ is statistically significant in dataset S1000.

The only exception to this trend is with dataset S1000 where $k = 1$ can lead to lower classification accuracy than $k \in \{2, 3\}$. However, only the difference between $k = 1$ and $k = 2$ is statistically significant ($p = 0.046$), while the difference between $k = 1$ and $k = 3$ is not ($p = 0.0549$).

Worth noting that when $k = 0$, k-skipped n-grams are identical to n-grams. Therefore the superiority of the classifiers when $k = 0$ suggests that n-grams can identify features that

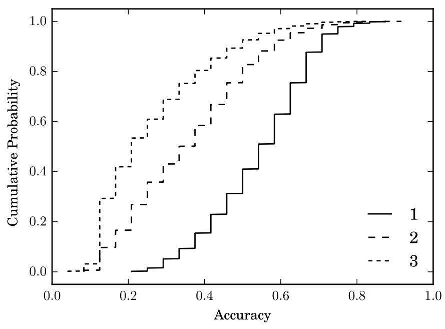

FIGURE 15. The ECDFs of classification accuracy from the perspective of the parameter *n* (S24 problems set).

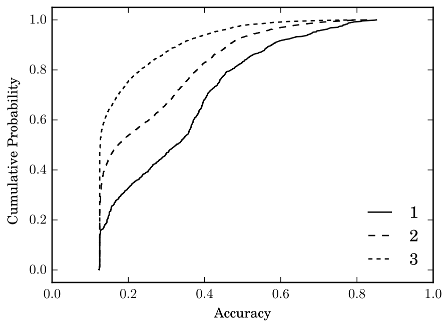

FIGURE 16. The ECDFs of classification accuracy from the perspective of the parameter *n* (S1000 problems set).

TABLE 11. *p* values of the most accurate methods from the perspective of the parameter *n* (S24 problems set).

n	2 acc $= 0.92$	3 acc $= 0.83$
1 acc $= 0.88$	$p = 0.7013$ $=$	$p = 0.7093$ $=$
2 acc $= 0.92$	–	$p = 0.3147$ $=$

lead to higher classification accuracy levels than can k-skip n-grams.

4) PARAMETER *n*

Figures 15 and 16 suggest that both datasets S24 and S1000 agree that bi-grams (i.e., $n = 2$) can identify of more accurate classification models than tri-grams (i.e., $n = 3$). While Table 11 shows that this observation is not statistically significant on dataset S24 ($p = 0.3147$), Table 12 shows that this is statistically significant in dataset S1000 ($p = 0.003$).

However, figures 15 and 16 disagree on the effectiveness of bi-grams over uni-grams (i.e., $n = 1$). Figure 15 suggests that bi-grams are also superior to uni-grams. However, this is not statistically significant on dataset S24. On the other hand, Figure 16 indicates the opposite while also showing statistical significance on dataset S1000.

5) PARAMETER GRAM

Figures 17 and 18 agree that the gram with the lowest upper bound limit on its accuracy is the `wordlen` gram (acc $=$ 0.67 in dataset S24 and acc $= 0.54$ in dataset S1000). Despite the small size of the dataset S24, our experiments show that the inferiority of `wordlen` against `pos` (the best performing

TABLE 12. *p* values of the most accurate methods from the perspective of the parameter *n* (S1000 problems set).

n	2 acc $= 0.83$	3 acc $= 0.80$
1 acc $= 0.85$	$p = 0.0480$ $*$	$p = 0.0010$ $***$
2 acc $= 0.83$	–	$p = 0.0030$ $**$

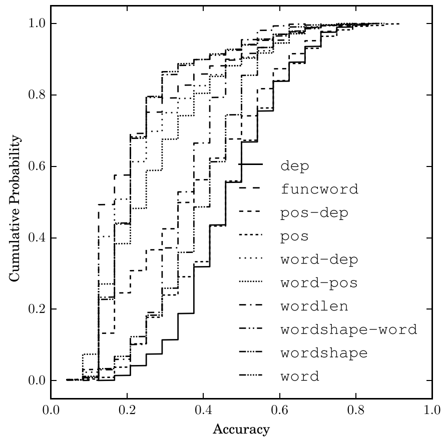

FIGURE 17. The ECDFs of classification accuracy from the perspective of grams (S24 problems set).

gram in S24) is statistically significant ($p = 0.036$). Interestingly, dataset S1000 shows that the inferiority of the gram `wordlen` against all other grams to be statistically highly significant ($p \leq 0.001$).

Figure 17 suggests that the gram that leads to the highest classification accuracy is `pos`, followed by `word`. However, dataset S24 (due to its size) cannot demonstrate statistical significance of any of the differences between the grams. The only exception where dataset S24 can show a statistically significant difference is concerning `wordlen`, as discussed earlier.

On the other hand, Figure 18 suggests that the gram that results in the highest classification accuracy is `word-dep`, followed closely by `word-pos`. Both of the grams have led to the same maximum classification accuracy of acc $= 0.85$.

The pattern that can be seen here is that generally, among both of the datasets, POS tag-based grams (`pos` in dataset S24 and `pos-word` in dataset S1000) tend to score one of the highest classification accuracy levels. Additionally, `pos` is the gram that allows for the second-highest classification accuracy levels in dataset S1000 with a statistically insignificant difference in accuracy against that of `word-pos` ($p =$ 0.0619).

B. DEPENDENT PARAMETERS EVALUATION

Section VIII-A presented the evaluation results from the perspective of the parameters l, `dir`, k, n, and grams, independently. This section aims to evaluate the features jointly by presenting clusters of the feature extraction methods, solely based on their classification accuracy, to answer some of the questions in Section VIII, specifically concerning the directions `spatial` and `deptree`.

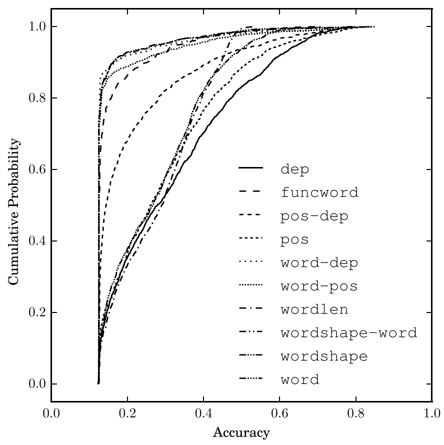

FIGURE 18. The ECDFs of classification accuracy from the perspective of grams (S1000 problems set).

To study the feature extraction methods, we first need to identify two clusters of such feature extraction methods: most accurate methods and least accurate methods. We identify the cluster of the most accurate feature extraction methods by following a principled approach as follows:

1) Obtain the most accurate feature extraction methods. In other words, all feature extraction methods that can achieve the maximum classification are identified.
2) Then, the list of the most accurate feature extraction methods is compared against all other feature extraction methods to identify other feature extraction methods that are similar enough to the most accurate methods. To identify such similar features in a principled manner, we perform pair-wise statistical significance tests between the most accurate methods and every other method in order to compute their p values. A feature extraction method is then considered to be similar enough if its difference in accuracy, against the most accurate ones, is not shown to be statistically significant ($p > 0.05$).

A similar approach is followed to identify the cluster of the least accurate feature extraction methods. In this case, the pair-wise statistical significance tests are performed against the least accurate feature extraction methods.

C. PARAMETERS EVALUATION RESULTS

By observing the list of the most accurate feature extraction methods, it can be seen that:

- The list is highly dominated by the `spatial` direction and the absence of the `deptree` direction. This suggests that syntactic n-grams with dependency trees are considerably limited in identifying accurate patterns to solve stylometry problems, such as the SCC author identification problem at hand.
- Feature extraction methods in the list also present small values of the parameter l. Feature extraction methods with small l suggest that they relatively identify reliable patterns that help the learning algorithm (RF) identify accurate classification models. However, this observation may differ depending on the noise presented in the dataset at hand, which in turn affects the degree by which noisy features are discarded.
- Values of the parameter k seem to be relatively diverse and not dominated by $k = 0$. This implies that k-skip n-grams as features are a step forward in identifying classification patterns that allow for more accurate stylometry problem solvers. Similarly, values of the parameter n are pretty diverse, which suggests the use of n-grams as a framework for identifying patterns that are suitable for solving stylometry problems.
- However, it is evident that in both of the datasets S24 and S1000, that $k = 0$ is dominant among the most accurate methods (acc $= 0.9167$ in S24, and acc $\in \{0.853, 0.851, 0.837\}$ in S1000). This suggests that classical n-grams, without the addition of k-skips as a parameter, have a slight edge. However, this is only shown to be statistically significant in the dataset S1000.

By observing the list of the least accurate feature extraction methods, it can be seen that the majority of the least accurate methods use the direction `deptree`. It is unclear whether the low classification accuracy is due to the direction parameter or whether it is due to the high values of l (which we have established earlier in Section VIII-A that larger values of l generally correspond to lower classification accuracy).

To isolate the effect of parameter l, we observe only the least accurate feature extraction methods that have small values of l (specifically, $l \leq 5$). Interestingly, the resultant filtered cluster of the least accurate feature extraction methods contains only those following the direction `deptree`. The only exception to this is only three features on the dataset S1000.

Therefore, it can be concluded that the distribution of uni grams, n-grams, and their generalization k-skip n-grams are effective feature extraction methods that allow to identify the most accurate SCC author identification models. However, syntactic n-grams with dependency trees identify features that lack considerably compared to the previous feature extraction methods.

IX. EVALUATION REPRODUCIBILITY

The feature extraction Python module is released in the repository https://gitlab.com/mmaakh/fextractor.git. The evaluation code (which makes use of the module `fextractor.py`) is released in the repository https://gitlab.com/mmaakh/stylometry-survey-evaluation.git. This repository contains two sub-directories: *evaluation* contains the code to generate the evaluation model outputs, and *visualization* contains the code to translate the model outputs into the figures and tables that are presented in Section VIII. The dependencies are Python,[3] Scikit-learn,[4] CoreNLP,[5] and Matplotlib.[6]

[3]https://python.org/
[4]http://scikit-learn.org/
[5]https://stanfordnlp.github.io/CoreNLP/
[6]http://matplotlib.org/

X. FUTURE DIRECTIONS

This section presents our thoughts about promising future research directions to address key challenges facing stylometry problem solvers.

A. IMPROVE OUR UNDERSTANDING OF THE SOLVERS

The best performing stylometry problem solvers are currently trained to maximize an optimization objective, such as the accuracy of the labels that they predict. However, there is almost no systematic component within the solvers that ensures that they are solving the problem that we think they are solving.

For example, an author attribution solver could be partially a topic detection model. Unless controlled datasets are used, or unless models are manually analyzed, it is unknown whether the solver is a style classifier, a topic classifier, or a mixture of both.

Future work may investigate the existence of algorithms with internals that ensure us that there is a systematic pressure on the model to be, say, an author attribution model. This is different than hoping that a generic learning algorithm would find the right model by analyzing some dataset, especially when the dataset is not controlled.

Historically, solvers used to analyze texts represented by a limited feature set, such as the distribution of their function words, to ensure that the solver is indeed analyzing texts' writing styles independent of their topics. However, such an approach is not ideal, as the distribution of function words in a given text contains some information about its topic. Additionally, such solvers tend to suffer degraded accuracy compared to state-of-the-art solvers.

A potentially promising approach could be via indirect training, by which a model is trained to solve a different problem than the target problem, such that the solver of the former problem is known beforehand to have knowledge of the solution of the target problem necessarily. The solver of the target problem is then extracted from the solver of the former problem. For example, Radford *et al.* found that, by training a large enough byte-level LSTM model for a month to solve the text review generation problem using a dataset comprised of millions of Amazon reviews, the model had indirectly also learned a state of art sentiment analyzer [56]. In other words, the review generator was learned in a supervised manner, while the indirectly learned sentiment analyzer within the review generator was learned in an unsupervised way. Similar effects are observed with other large neural networks as their hidden neurons tend to indirectly learn solvers of intermediate problems related to solving the target problem.

It would be interesting to evaluate a similar setup with stylometry-problem solvers to learn a writing style detector in an unsupervised manner indirectly. For example, a large enough recurrent model that is trained for solving the text summarization problem would possibly also indirectly learn to identify portions of texts that concern topics discussed in the input texts (so that the summarizer model considers them for inclusion in its output summary), and parts of texts that concern the writing style of its author (so that it excludes it from its summary). This may imply the existence of an indirectly-learned topic-style classifier, in an unsupervised manner, within such text summarization models.

Suppose the text summarization model is accurate for its purpose (summarizing texts). In that case, it should follow that its intermediately-learned style detection is accurate also (so that it avoids including the irrelevant information about writing styles in its summaries). Therefore, if a stylometry problem solver used this intermediately-learned style detector, we may have an additional reason to believe that it is indeed deciding based on writing styles. In other words, it might be possible to evaluate the correctness of such a solver not only based on its performance against some evaluation sets but also based on the performance of the text summarization model.

B. IMPROVE SOLVERS' ROBUSTNESS AGAINST ADVERSARIAL ATTACKS

Stylometry problem solvers are generally weak against adversarial attacks, such as obfuscation or imitation attacks, by which classification accuracy approaches random chance guessing (with obfuscation attacks) and below random chance guessing (with imitation attacks).

Future work may explore the feasibility of training stylometry problem solvers using Generative Adversarial Network (GAN) frameworks, by which a stylometry problem solver is not only trained to predict the correct label for the given problems but also trained not to be fooled by another competing model that is also being trained with it, except for having the opposing task of fooling the solver (for example, by manipulating input texts to imitate other authors).

If successful, such an approach would have the benefit of producing stylometry problem solvers that are systematically constructed to have increased robustness against adversarial attacks in an unsupervised manner.

C. REDUCE SOLVERS' RUN-TIME

Currently, some of the most accurate AV solvers rely on obtaining language models for authors by training deep neural networks, which also makes some of the slowest problem solvers due to the expensive training time that such deep neural networks require. This significantly limits their applicability to scenarios with large numbers of candidate authors.

An interesting future work might be to analyze the internals of many such expensively-trained deep neural network language models to possibly identify common functions, such as high-level feature-extraction functions, that the networks aim to learn.

Once the common functions are identified, investigate methods to re-use them in the training phase of the future deep neural network language models so that the common functions are not re-learned for every language model from scratch every time a new unknown text is analyzed.

This could be taken one step further by implementing the surveyed feature extraction methods as differentiable functions. This could allow us to re-use them in future neural networks without losing the end-to-end differentiability of the network's error function. Since the differentiable feature extraction functions are optimized beforehand, future neural networks that make use of them would only require optimizing the network's weights, excluding those of the functions.

XI. CONCLUSION

This paper introduced electronic text stylometry problems under a unified notation in probability terms, their importance in enhancing various upper-layer applications, the key challenges currently faced in this field, the critical limitations of stylometry problem solvers, and suggestions for future directions to solve them. Such challenges include optimizing the stylometry problem solvers to maximize their classification accuracy and performing accurate stylometry analysis across distinct domains. The challenges also include the generalization of existing data representation methods to enhance our understanding.

This paper has also addressed a critical challenge by generalizing many feature extraction functions as special cases of the *at least l-frequent* `dir`*-directed k-skipped n-grams*, as well as presenting an extensive evaluation of over tens of thousands of feature-extraction functions, which evaluated them under the same unified testing bed. This allowed us to perform the first comparisons between previously proposed feature extraction functions in the literature (e.g., comparing syntactic n-grams against k-skipped n-grams) and to introduce novel definitions of grams (e.g., compound grams). Interestingly, despite the diversity of the set of the feature extraction functions, classical n-gram-based functions proved to be superior among the more sophisticated variants (i.e., syntactic n-grams with dependency trees and k-skipped n-grams). Our future work will focus on using transfer learning to reduce solvers' run-time and tackle multi-domain authorship identification by reusing a model trained in a particular domain as a starting point for author identification in another domain.

REFERENCES

[1] L. Fridman, S. Weber, R. Greenstadt, and M. Kam, "Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location," *IEEE Syst. J.*, vol. 11, no. 2, pp. 513–521, Jun. 2016.

[2] N. Pokhriyal, K. Tayal, I. Nwogu, and V. Govindaraju, "Cognitive-biometric recognition from language usage: A feasibility study," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 134–143, Jan. 2017.

[3] M. Abuhamad, A. Abusnaina, D. Nyang, and D. Mohaisen, "Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 65–84, Jan. 2021.

[4] R. Igawa, A. Almeida, and B. Zarpelão, "Recognition of compromised accounts on Twitter," in *Proc. Anais do Simpósio Brasileiro de Sistemas de Informação (SBSI)*. Porto Alegre, Brazil: Brazilian Computer Society, May 2015, pp. 2:9–2:14.

[5] S. Kanwal, S. Nawaz, M. K. Malik, and Z. Nawaz, "A review of text-based recommendation systems," *IEEE Access*, vol. 9, pp. 31638–31661, 2021.

[6] W. Daelemans, "Explanation in computational stylometry," in *Proc. 14th CICLing*, vol. 2. Berlin, Germany: Springer-Verlag, 2013, pp. 451–462.

[7] D. Kernot, T. Bossomaier, and R. Bradbury, "The stylometric impacts of ageing and life events on identity," *J. Quant. Linguistics*, vol. 26, no. 1, pp. 1–21, Dec. 2017.

[8] P. Bora, T. Awalgaonkar, H. Palve, R. Joshi, and P. Goel, "ICodeNet—A hierarchical neural network approach for source code author identification," in *Proc. 13th Int. Conf. Mach. Learn. Comput. (ICMLC)*. New York, NY, USA: ACM, 2021, pp. 180–185, doi: 10.1145/3457682.3457709.

[9] V. Kalgutkar, R. Kaur, H. Gonzalez, N. Stakhanova, and A. Matyukhina, "Code authorship attribution: Methods and challenges," *ACM Comput. Surveys*, vol. 52, no. 1, pp. 1–36, Feb. 2019, doi: 10.1145/3292577.

[10] A. C. Islam, F. Yamaguchi, E. Dauber, R. E. Harang, K. Rieck, R. Greenstadt, and A. Narayanan, "When coding style survives compilation: De-anonymizing programmers from executable binaries," *CoRR*, vol. abs/1512.08546, pp. 1–15, Dec. 2015.

[11] R. Sarwar, N. Urailertprasert, N. Vannaboot, C. Yu, T. Rakthanmanon, E. Chuangsuwanich, and S. Nutanong, "CAG: Stylometric authorship attribution of multi-author documents using a co-authorship graph," *IEEE Access*, vol. 8, pp. 18374–18393, 2020.

[12] N. Potha and E. Stamatatos, "Improved algorithms for extrinsic author verification," *Knowl. Inf. Syst.*, vol. 62, no. 5, pp. 1903–1921, May 2020.

[13] L. Ouyang, Y. Zhang, H. Liu, Y. Chen, and Y. Wang, "Gated POS-level language model for authorship verification," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 4025–4031.

[14] O. Halvani, L. Graner, and R. Regev, "TAVeer: An interpretable topic-agnostic authorship verification method," in *Proc. 15th Int. Conf. Availability, Rel. Secur.*, Aug. 2020, pp. 41:1–41:10.

[15] M. Khonji and Y. Iraqi, "Evaluating author attribution on Emirati tweets," *IEEE Access*, vol. 8, pp. 149531–149543, 2020.

[16] P. Juola, "Authorship attribution," *Found. Trends Inf. Retr.*, vol. 1, no. 3, pp. 233–334, Dec. 2006.

[17] M. Koppel, J. Schler, and S. Argamon, "Computational methods in authorship attribution," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 60, no. 1, pp. 9–26, Jan. 2009.

[18] E. Stamatatos, "A survey of modern authorship attribution methods," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 60, no. 3, pp. 538–556, Mar. 2009.

[19] N. Potha and E. Stamatatos, "Improving author verification based on topic modeling," *J. Assoc. Inf. Sci. Technol.*, vol. 70, no. 10, pp. 1074–1088, Oct. 2019.

[20] B. T. Boenninghoff, J. Rupp, R. M. Nickel, and D. Kolossa, "Deep Bayes factor scoring for authorship verification," in *Proc. Conf. Labs Eval. Forum (CLEF)*, vol. 2696, Sep. 2020, pp. 1–12.

[21] B. Boenninghoff, S. Hessler, D. Kolossa, and R. M. Nickel, "Explainable authorship verification in social media via attention-based similarity learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 36–45.

[22] E. Castillo, O. Cervantes, and D. Vilariño, "Authorship verification using a graph knowledge discovery approach," *J. Intell. Fuzzy Syst.*, vol. 36, no. 6, pp. 6075–6087, Jun. 2019.

[23] P. Joula and E. Stamatatos, "Overfiew of the author identification tasK at PAN 2013," in *Proc. Conf. Labs Eval. Forum*, 2013, pp. 1–20.

[24] E. Stamatatos, W. Daelemans, B. Verhoeven, B. Stein, M. Potthast, P. Juola, M. A. Sánchez-Pérez, and A. Barrón-Cede no, "Overview of the author identification task at PAN 2014," in *Proc. CLEF*, Sheffield, U.K., Sep. 2014, pp. 1–21.

[25] E. Stamatatos, W. Daelemans, B. Verhoeven, P. Juola, A. López-López, M. Potthast, and B. Stein, "Overview of the author identification task at PAN 2015," in *Proc. Eval. Labs Workshop-Work. Notes Papers (CLEF)*, Toulouse, France, Sep. 2015, pp. 1–17.

[26] G. U. Yule, *The Statistical Study of Literary Vocabulary*. Cambridge, U.K.: Cambridge Univ. Press, 1944.

[27] A. Honore, "Some simple measure of richness of vocabulary," *Assoc. Literary Linguistic Comput. Bull.*, vol. 7, no. 2, pp. 172–177, 1979.

[28] K. Tanaka-Ishii and S. Aihara, "Computational constancy measures of Texts—Yule's K and Rényi's entropy," *Comput. Linguistics*, vol. 41, no. 3, pp. 481–502, Sep. 2015.

[29] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic dependency-based N-grams as classification features," in *Advances in Computational Intelligence*. Berlin, Germany: Springer, 2013, pp. 1–11.

[30] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

[31] D. Bagnall, "Author identification using multi-headed recurrent neural networks," in *Proc. Eval. Labs Workshop-Work. Notes Papers (CLEF)*, Toulouse, France, Sep. 2015, pp. 1–11.

[32] M. Koppel, J. Schler, and E. Bonchek-Dokow, "Measuring differentiability: Unmasking pseudonymous authors," *J. Mach. Learn. Res.*, vol. 8, pp. 1261–1276, Jun. 2007.

[33] A. Ramponi and B. Plank, "Neural unsupervised domain adaptation in NLP—A survey," *CoRR*, vol. abs/2006.00632, pp. 1–18, May 2020. [Online]. Available: https://arxiv.org/abs/2006.00632

[34] O. de Vel, A. Anderson, M. Corney, and G. Mohay, "Mining e-mail content for author identification forensics," *ACM SIGMOD Rec.*, vol. 30, no. 4, pp. 55–64, Dec. 2001.

[35] J. Diederich, J. Kindermann, E. Leopold, and G. Paass, "Authorship attribution with support vector machines," *Appl. Intell.*, vol. 19, nos. 1–2, pp. 109–123, 2003, doi: 10.1023/A:1023824908771.

[36] J. Li, R. Zheng, and H. Chen, "From fingerprint to writeprint," *Commun. ACM*, vol. 49, no. 4, pp. 76–82, Apr. 2006.

[37] C. Sanderson and S. Guenter, "Short text authorship attribution via sequence kernels, Markov chains and author unmasking: An investigation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2006, pp. 482–491.

[38] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard, "Surveying stylometry techniques and applications," *ACM Comput. Surveys*, vol. 50, no. 6, pp. 1–36, Jan. 2018, doi: 10.1145/3132039.

[39] X. Tang, S. Liang, and Z. Liu, "Authorship attribution of the golden lotus based on text classification methods," in *Proc. 3rd Int. Conf. Innov. Artif. Intell. (ICIAI)*. New York, NY, USA: ACM, 2019, pp. 69–72, doi: 10.1145/3319921.3319958.

[40] M. Khonji, Y. Iraqi, and A. Jones, "An evaluation of authorship attribution using random forests," in *Proc. Int. Conf. Inf. Commun. Technol. Res. (ICTRC)*, May 2015, pp. 68–71.

[41] J. Fréry, C. Largeron, and M. Juganaru-Mathieu, "UJM at CLEF in author identification," in *Proc. Eval. Labs Workshop-Work. Notes Papers (CLEF)*, Sheffield, U.K., Sep. 2014, pp. 1–7.

[42] R. A. J. Matthews and T. V. N. Merriam, "Neural computation in stylometry I: An application to the works of Shakespeare and Fletcher," *Literary Linguistic Comput.*, vol. 8, no. 4, pp. 203–209, 1993.

[43] T. V. N. Merriam and R. A. J. Matthews, "Neural computation in stylometry II: An application to the works of Shakespeare and Marlowe," *Literary Linguistic Comput.*, vol. 9, no. 1, pp. 1–6, Jan. 1994.

[44] F. J. Tweedie, S. Singh, and D. I. Holmes, "Neural network applications in stylometry: The federalist papers," *Comput. Humanities*, vol. 30, no. 1, pp. 1–10, 1996, doi: 10.1007/BF00054024.

[45] F. Khosmood and R. Levinson, "Toward unification of source attribution processes and techniques," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Aug. 2006, pp. 4551–4556.

[46] V. Keselj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for authorship attribution," in *Proc. PACL*, 2003, pp. 255–264.

[47] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas, "Source code author identification based on N-gram author profiles," in *Proc. 3rd IFIP Conf. Artif. Intell. Appl. Innov.*, Athens, Greece. Boston, MA, USA: Springer, 2006, pp. 508–515.

[48] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proc. 10th ACM SIGKDD*. New York, NY, USA: ACM, 2004, pp. 206–215.

[49] R. Cilibrasi and P. M. B. Vitányi, "Clustering by compression," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1523–1545, Apr. 2005.

[50] S. Evert, T. Proisl, F. Jannidis, I. Reger, S. Pielstrom, C. Schoch, and T. Vitt, "Understanding and explaining Delta measures for authorship attribution," *Digit. Scholarship Humanities*, vol. 32, no. 2, pp. II4–II16, Jun. 2017, doi: 10.1093/llc/fqx023.

[51] M. Koppel and S. Seidman, "Automatically identifying pseudepigraphic texts," in *Proc. EMNLP*, 2013, pp. 1449–1454.

[52] M. Khonji and Y. Iraqi, "A slightly-modified GI-based author-verifier with lots of features (ASGALF)," in *Proc. Eval. Labs Workshop Work. Notes Papers (CLEF)*, Sheffield, U.K., Sep. 2014, pp. 1–7.

[53] M. Brennan, S. Afroz, and R. Greenstadt, "Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 3, pp. 1–22, Nov. 2012, doi: 10.1145/2382448.2382450.

[54] Y. Seroussi, I. Zukerman, and F. Bohnert, "Authorship attribution with latent Dirichlet allocation," in *Proc. 15th CoNLL*. Stroudsburg, PA, USA: ACL, 2011, pp. 181–189.

[55] P. Juola, "An overview of the traditional authorship attribution subtask," in *Proc. Eval. Labs Workshop-Work. Notes Papers (CLEF)*, Rome, Italy, Sep. 2012, pp. 1–7.

[56] A. Radford, R. Józefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *CoRR*, vol. abs/1704.01444, pp. 1–9, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1704.01444

MAHMOUD KHONJI (Member, IEEE) received the master's (research) and Ph.D. degrees in engineering from Khalifa University, in 2012 and 2017, respectively. In Khalifa University, he worked on various text classification problems relating to security, forensics, and privacy. His research interests include security, privacy, and machine learning.

YOUSSEF IRAQI (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Montreal, Canada, in 2000 and 2003, respectively. He is currently an Associate Professor with the School of Computer Science, Mohammed VI Polytechnic University, Morocco. Before that, he was with the Department of Electrical Engineering and Computer Science, Khalifa University, United Arab Emirates, for 12 years. Before joining KU, he was the Chair of the Department of Computer Science, Dhofar University, Oman, for four years. From 2004 to 2005, he was a Research Assistant Professor with the David R. Cheriton School of Computer Science, University of Waterloo, Canada. He has published more than 110 research articles in international journals and refereed conference proceedings. His research interests include resource management in wireless networks, blockchain, trust and reputation management, cloud computing, and stylometry. In 2008, he received the IEEE Communications Society Fred W. Ellersick Paper Award in the field of communications systems. He is on many technical program committees of international conferences and always approached for his expertise by international journals in his field.

LOUBNA MEKOUAR (Senior Member, IEEE) received the M.Sc. degree in computer science from the University of Montreal, Canada, in 1999, and the Ph.D. degree in computer science from the University of Waterloo, Canada, in 2010. She is currently an Assistant Professor with the School of Computer Science, Mohammed VI Polytechnic University, Morocco. Before that, she was an Assistant Professor with the College of Technological Innovation, Zayed University, United Arab Emirates, for three years. Her research interests include trust and reputation management, distributed systems, blockchain, and recommender systems.