Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Article

De-anonymizing Authors of Electronic Texts: A Survey on Electronic Text Stylometry

Mahmoud Khonji [1] and Youssef Iraqi [2]

[1] Khalifa Univerity; m@khonji.org
[2] Khalifa Univerity; youssef.iraqi@ku.ac.ae

Abstract: Electronic text stylometry is a collection of forensics methods that analyze the writing styles of input electronic texts in order to extract information about authors of the input electronic texts. Such extracted information could be the identity of the authors, or aspects of the authors, such as their gender, age group, ethnicity, etc. This survey paper presents the following contributions: 1) A description of all stylometry problems in probability terms, under a unified notation. To the best of our knowledge, this is the most comprehensive definition to date. 2) A survey of key methods, with a particular attention to data representation (or feature extraction) methods. 3) An evaluation of 23,760 feature extraction methods, which is the most comprehensive evaluation of feature extraction methods in the literature of stylometry to date. The importance of this evaluation is two fold. First, identifying the relative effectiveness of the features (since, currently, many are not evaluated jointly; e.g. syntactic n-grams are not evaluated against k-skip n-grams, and so forth). Second, thanks to our generalizations, we could evaluate novel grams, such as what we name *compound grams*. 4) The release of our associated Python feature extraction library, namely *Fextractor*. Essentially, the library generalizes all existing n-gram based feature extraction methods under the *at least l-frequent*, `dir`*-directed, k-skipped n-grams*, and allows grams to be diversely defined, including definitions that are based on high-level grammatical aspects, such as Part of Speech (POS) tags, as well as lower-level ones, such as distribution of function words, word shapes, etc. This makes the library, by far, the most extensive in this domain to date. 5) The construction, evaluation, and release of the first dataset for Emirati social media text. This evaluation represents the first evaluation of author identification against Emirati social media texts. Interestingly, we find that, when using our models and feature extraction library (Fextractor), authors could be identified significantly more accurately than what is reported with similarly sized datasets. The dataset also contains sub-datasets that represent other languages (Dutch, English, Greek and Spanish), and our findings are consistent across them.

Keywords: stylometry; author identification; author verification; author profiling; stylistic inconsistency; text analysis; supervised learning; unsupervised learning; classification; forensics

1. Introduction

Improving solvers of stylometry problems is important for enhancing various application domains, such as forensics, privacy (or anti-forensics), active-authentication [1–3], compromised account detection [4], recommender systems [5], deception detection, market analysis, and medical diagnosis [6]. Author identification can also be accurately performed on program source codes [7] as well as compiled binaries [8]. 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 certain illnesses. 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 that are 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), age group (e.g. 10s, 20s, 30s, etc), ethnicity group (e.g. the most popular ethnicity groups in the UAE are (ordered alphabetically) Bangladeshi, Emirati, Filipino, Indian, Pakistani, etc). The target classification label is "*male*" or "*female* in the case of gender detection, "*10s*", "*20s*", "*30s*", etc in the case of age-group detection, and so forth.
- Author Diarization (AD): given a single text whose authors are unknown, cluster its parts such that each cluster only has parts that are written by the same author as every other text in the same cluster. 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:

- Target information: authors' identities (AA, AC, AD, AV), or their profile attributes (AP).
- Problem assumptions: whether the classification model is expected to handle situations where the answer is "*else*". If the model is expected to handle situations where the answer is "*else*", 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 else other than the known author. This expectation increases the difficulty of the problem as the model is not only expected to model known authors (e.g. Shakespeare), but also any other author (e.g. anyone that is not Shakespeare).

  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. This is due to the fact that AA expects that the solution author 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.

Worth noting that it is possible to model open-set problems as closed-set problems. For example, modeling the AV problem as a closed-set binary-classification problem of "*yes*" and "*no*". However, this *binarization* of the AV problems a known baseline in the literature and is found to yield inferior classification accuracy relative to the case when AV problems are treated as open-set classification problems [9].

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

- AV problems are known as the fundamental problem of stylometry. This is due to the fact that the other stylometry problems can be decomposed into a set of AV problems [6,10–12].

- Due to the open-set nature of AV problems and their solvers, they have a broader application domain than the closed-set stylometry problems. For example, in realistic situations it is not uncommon for a questioned test text to be authored by an individual that none of his/her texts are seen in the learning phase. On the other hand, AA problem solvers expect that the author of all input test texts is necessarily one that has also authored texts that exist in the learning set. This causes AA problem solvers to return a classification label that is necessarily incorrect when this assumption is violated. However, AV problem solvers are not bound to this, and therefore, AV problems are more realistic problems than AA 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.

Table 1 presents a summary of categories of all stylometry problems from the perspective of the target information that they seek to infer, and the problem assumptions as followed by the literature.

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

As presented in Table 1, inferring the identity of authors of input texts is modeled in the literature as both open and closed-set problems. However, inferring the profile attributes of authors of input texts is —so far— modeled in the literature as closed-set problems, only.

Another fundamental aspect of electronic stylometry problem solvers is the methods that are used to represent input electronic texts. One of such methods is simply using the raw texts. Other methods could represent texts as bags of words, or vectors. One of the key contributions of this survey is an extensive evaluation of many such feature extractions functions. This extensive evaluation is particularly important due to the fact that the known data representation methods are often not evaluated under the same unified test bed.

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. network) that an author used to publish his/her works, then one can use deterministic methods to track the source of the text publication, ultimately leading to the author. Gaining access to such infrastructure can be achieved by having legitimate administrative privileges for legalized interception, or illegitimately via the use of malware, back-doors, or by taking advantage of network anonymization techniques (e.g. Tor networks).

What sets the stylometric methods (focus of this paper) apart from the non-stylometric ones, is the fact that the former can be executed without the need of requiring access to the underlying messaging infrastructure (let it be legitimately or illegitimately). Because of this, stylometric author identification methods can be applicable in cases where the non-stylometric methods fail to apply, such as the case when obtaining adequate access to the messaging infrastructures is not feasible. In other words, while the problem domains of the stylometric and non-stylometric methods overlap, there remains problems that are only solvable by the former.

Additionally, in scenarios where author identification problems are solvable by, both, stylometric and non-stylometric methods, the stylometric methods can still be used to provide further evidence to enhance the solution to the author identification problem.

This paper is structured as follows: Section 2 presents some of the most significant challenges that face today's state of electronic text stylometry. Section 3 defines a number of fundamental classification problems that can be used to model all stylometry problem solvers. Section 4 presents

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

a comprehensive definition of text representation methods, as commonly used in the literature, in addition to our generalizations. Section 5 introduces our extensive feature extraction library, Fextractor. Section 6 defines all stylometry problems known to date. Section 7 introduces a number of notable stylometry problem solvers. Section 8 presents our evaluation methodology of the extensive set of feature extraction functions, followed by the evaluation results in Section 9. Section 10 presents the Emirati social media author identification dataset, and our evaluation methodology. Followed by the evaluation results in Section 11. Section 12 draws the concluding remarks.

2. Challenges

To the best of our knowledge, the most significant challenges that face electronic text stylometry problems are centred around identifying the classification accuracy of existing methods, as well as enhancing their 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 in achieving more accurate stylometry problem solvers. Therefore, it is critical to question the various aspects of state of the art stylometry problem solver methods, for the purpose of identifying such parameters, and their alternative variants.
- Cross-domain stylometry: author identification problem solvers tend to classify a pair of texts that each of them falls under a distinct domain (e.g. distinct topics, genres, times, etc) to be written by distinct authors, even when they are not, This is due to the fact that domain dissimilarity affects all similarity measures up to an extinct that can negatively impact the accuracy of author identification models. 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 is a key limitation in the failure of accurately solving AV problems in Big Data scenarios as such data is often characterized by its high variety.
- Generalization of existing data representation methods: many methods of representing electronic texts (or feature extraction methods) are proposed, however the current state of the literature on stylometry seems to lack adequate generalizations for such methods. This, effectively, reduces our ability in identifying novel variants of existing methods and feature extraction functions.
- Software availability: lack of conveniently-available, and extensive, software that implement the many existing stylometry methods and feature extraction functions. There is often a tremendous need in re-developing the many proposed methods or functions, and because of the sheer amount of effort that is required to develop as such, it is common that most of the methods or functions are not adequately evaluated. As a result, the true value of the numerous independent contributions, relative to the each other, is often not adequately known.
- Evaluation datasets: lack of evaluation datasets for stylometry problem solvers, when executed against electronic texts that are written in Emirati Arabic, a dialect of the Arabic language that is natively spoken in the United Arab Emirates (UAE). This effectively casts uncertainty with respect to the performance of all stylometry methods, when evaluated against electronic texts that are written in this dialect. As a result, the applicability of electronic text stylometry methods against Emirati texts for the purpose of enhancing forensics, anti-forensics or market analysis, is unknown.

This survey paper moves towards addressing the last three challenges, namely: genelizing existing data representation methods, releasing our extensive feature extraction library under a premising open-source license, as well as the construction and the release of our Emirati tweets dataset.

3. Fundamental Classification Problems

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

All stylometry problems can be classified in one of the following categories: 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.

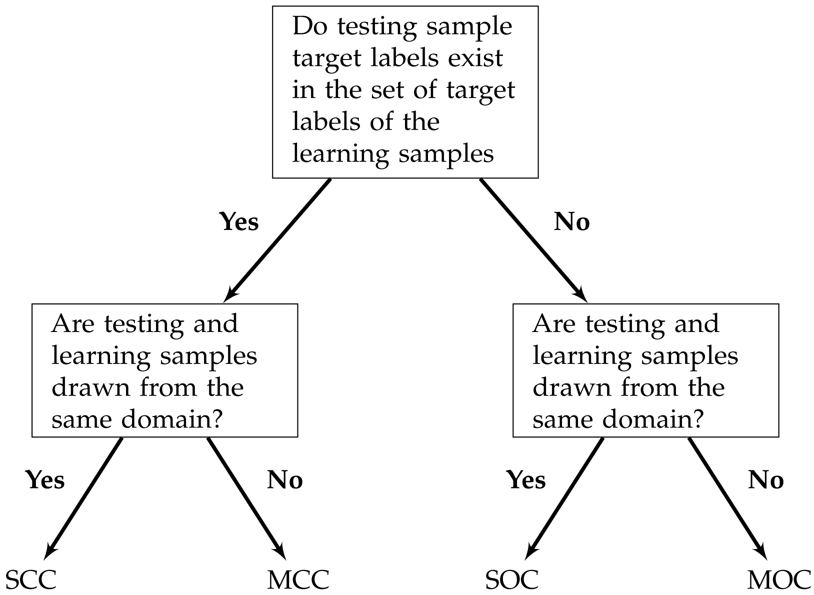

Figure 1. A categorization of fundamental stylometry problems.

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

3.1. Notation

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

- 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, genres, etc).
- Let $\mathcal{Q}$ be the index set of all classification tasks (author identification, gender identification, age-group identification, etc).
- 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, time of authorship, etc.
  - $\mathbf{x}_{i,d}$ 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, then the labels represent author identifiers. On the other hand, if the task is AP for the purpose of gender detection, then $y_{i,q} \in \{\text{male}, \text{female}\}$.
- fex is a function that represents texts as dim dimensional vectors in $\mathbb{R}^{\text{dim}}$:

$$\begin{aligned} \text{fex} : \{x_{i,d} : i \in \mathcal{I}, d \in \mathcal{D}\} &\to \mathbb{R}^{\text{dim}} \\ x_{i,d} &\mapsto \mathbf{x}_{i,d} \end{aligned} \tag{1}$$

- $\mathcal{X}_d = \{\mathbf{x}_{i,d} : i \in \mathcal{I}\}$ 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,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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, that estimates their value should they have been written by a process that has 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:

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

3.2. 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:

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

However, what identifies a classification model as an SCC is its input as well as 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. In other words, differences between the values of the represented text samples is not because of differences of their classification labels under unrelated classification tasks $\mathcal{Q} \setminus q$. The differences (if any) are because of non-systematic differences (i.e. random chance). The reason for this is due to the fact that if there was a systematic difference between samples with same lables under the same classification task q, such difference would be associated with a different classification task than q. The lack of lable difference with other classification tasks than q implies that there is no systematic difference other than what is associated with the labels under the task q.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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:

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

3.3. 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 that of 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 $y \in \mathcal{Y}_{T,q}$,*

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

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

Therefore, MCC problem solvers can be modeled as follows:

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

3.4. 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).

Formally, let $\mathcal{X}_{d,y_{i,q},1} \subseteq \mathcal{X}_{d,y_{i,q}}$ and $\mathcal{X}_{d,y_{j,q},2} \subseteq \mathcal{X}_{d,y_{j,q}}$ be two subsets that contain represented texts that correspond to classification labels $y_{i,q}$ and $y_{j,q}$, respectively, such that $\mathcal{X}_{d,y_{i,q},1} \cup \mathcal{X}_{d,y_{j,q},2} \in \mathcal{X}_{d,T}$, and $\mathcal{X}_{d,y_{i,q},1} \cap \mathcal{X}_{d,y_{j,q},2} = \emptyset$ (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 $y_{i,q} = y_{j,q}$. This can be answered in probability terms as follows:

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

where $Y_{T,q,1}$ and $Y_{T,q,2}$ are independent random variables that take values in $\mathcal{Y}_{T,q}$, $X_{d,T,1}$ and $X_{d,T,2}$ are independent random variables that take values in $\mathcal{X}_{d,T}$, 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:

$$\Pr(Y_{q,1} = Y_{q,2} | X_{d,L,1} = \mathcal{X}_{d,y_{i,q},1}, X_{d,L,2} = \mathcal{X}_{d,y_{j,q},2}) + \epsilon \tag{8}$$

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

$$\begin{cases} \text{Yes, } y_{i,q} = y_{j,q} & \text{if } \Pr(Y_{q,1} = Y_{q,2} | X_{d,L,1} = \mathcal{X}_{d,y_{i,q},1}, X_{d,L,2} = \mathcal{X}_{d,y_{j,q},2}) > 0.5 \\ \text{No, } y_{i,q} \neq y_{j,q} & \text{otherwise} \end{cases} \tag{9}$$

However, since SOC problems do not assume that $\mathcal{Y}_{q,T} \subseteq \mathcal{Y}_{q,L}$, 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 [13–15]) to strictly define $\mathcal{Y}_{q,T} \cap \mathcal{Y}_{q,L} = \emptyset$. This is to ensure 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 $\mathcal{Y}_{q,L}$).

In order 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 $\mathcal{X}_{d,y_{1,q},1} \subseteq \mathcal{X}_{d,y_{1,q}}$, $\mathcal{X}_{d,y_{2,q},2} \subseteq \mathcal{X}_{d,y_{2,q}}$, $\mathcal{X}_{d,y_{3,q},3} \subseteq \mathcal{X}_{d,y_{3,q}}$ and $\mathcal{X}_{d,y_{4,q},4} \subseteq \mathcal{X}_{d,y_{4,q}}$, where we know beforehand that $y_{1,q} = y_{2,q}$, $y_{3,q} = y_{4,q}$, but $y_{1,q} \neq y_{3,q}$. Additionally, let $X_{d,y_{1,q},1}$, $X_{d,y_{2,q},2}$, $X_{d,y_{3,q},3}$ and $X_{d,y_{4,q},4}$ be random variables that take values in the subsets, respectively. Furthermore, suppose that their PDFs visualize as presented in Figure 2.

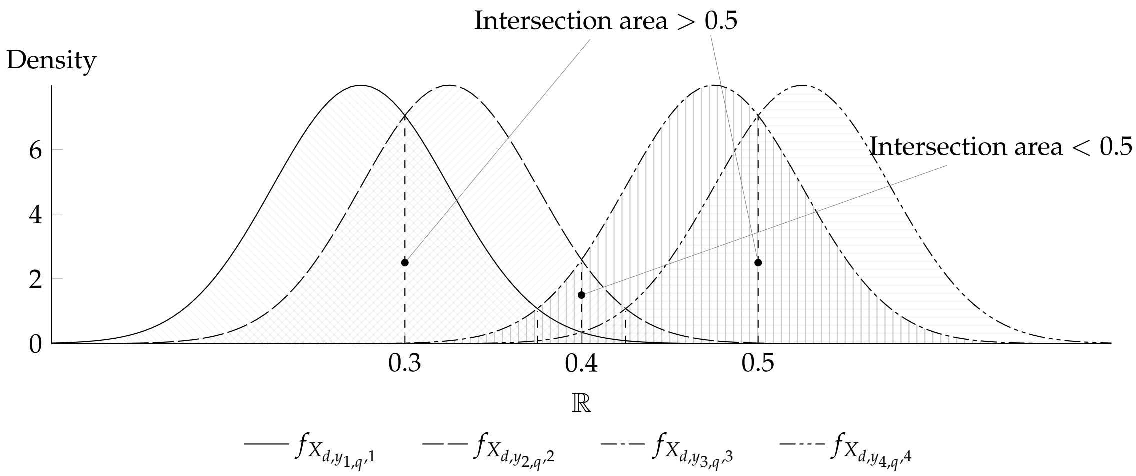

Figure 2. Hypothetical examples of the PDFs $f_{X_{d,y_{1,q},1}}$, $f_{X_{d,y_{2,q},2}}$, $f_{X_{d,y_{3,q},3}}$ and $f_{X_{d,y_{4,q},4}}$.

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 clearly over-fit the specific learning subsets in $\mathcal{X}_{d,L}$ as the subsets represent authors that have their represented texts to be centered around 0.3 and 0.5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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, it happens that subset pairs that share the same classification label, regardless of the value of the label, also share an intersection area that is greater than 0.5.

Worth noting that the SOC problems are often referred to in the literature as fundamental problem of stylometry [6]. This is due to the fact that all stylometry problems can be broken into a set of binary decision problems that, if solved correctly, would indeed lead into solving the initial problem. A relatively straight-forward use case of the SOC problem is the clustering problem, by which sets of texts are exhaustively paired into many SOC problems, and then clustered based on their pair-wise similarity.

3.5. 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 due to the fact that there is a domain mismatch between samples of the learning set, and samples of the testing set.

Formally, let $\mathcal{X}_{b,y_{i,q},1} \subseteq \mathcal{X}_{b,y_{i,q}}$ and $\mathcal{X}_{b,y_{j,q},2} \subseteq \mathcal{X}_{b,y_{j,q}}$ be two subsets that contain represented texts that correspond to classification labels $y_{i,q}$ and $y_{j,q}$, respectively, such that $\mathcal{X}_{b,y_{i,q},1} \cup \mathcal{X}_{b,y_{j,q},2} \in \mathcal{X}_{b,T}$, and $\mathcal{X}_{b,y_{i,q},1} \cap \mathcal{X}_{b,y_{j,q},2} = \emptyset$. Then, similar to SOC problems, the MOC problem at hand is to infer whether $y_{i,q} = y_{j,q}$. 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 domains 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?

In order 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:

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

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

In order to visualize this transformation by the DA function, consider the following hypothetical example of the following subsets of represented texts $\mathcal{X}_{d,y_{1,q},1} \subseteq \mathcal{X}_{d,y_{1,q}}$ (note that its domain is d), $\mathcal{X}_{b,y_{2,q},2} \subseteq \mathcal{X}_{b,y_{2,q}}$ and $\mathcal{X}_{b,y_{3,q},3} \subseteq \mathcal{X}_{b,y_{3,q}}$ (note that the domain of the latter subsets is b), where we know beforehand that $y_{1,q} = y_{3,q}$, but $y_{1,q} \neq y_{2,q}$. Additionally, let $X_{d,y_{1,q},1}$, $X_{b,y_{2,q},2}$ and $X_{b,y_{3,q},3}$ 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

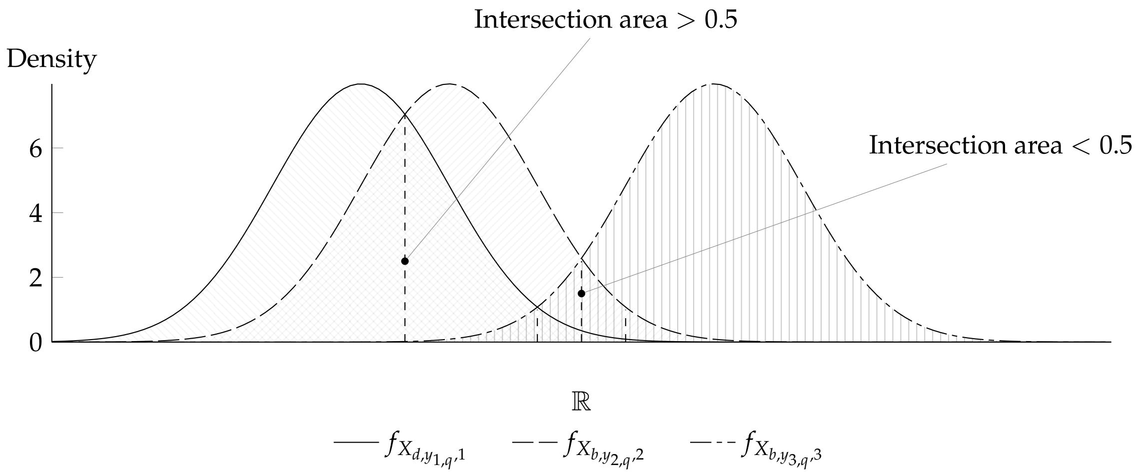

Figure 3. Hypothetical examples of the PDFs $f_{X_{d,y_{1,q}},1}$, $f_{X_{b,y_{2,q}},2}$ and $f_{X_{b,y_{3,q}},3}$, before applying the DA function $z_{b \to d}$.

294 Note that in the example in Figure 3, the intersection area between the PDFs $f_{X_{d,y_{i,q}},1}$ and $f_{X_{d,y_{2,q}},2}$ is greater than 0.5, despite the fact that they do not share the same classification label (i.e. $y_{1,q} \neq y_{2,q}$). On the other hand, the intersection area between the PDFs $f_{X_{d,y_{i,q}},1}$ and $f_{X_{d,y_{2,q}},3}$ is less than 0.5, despite the fact that they share the same classification label (i.e. $y_{1,q} = y_{3,q}$). This is possibly due to the fact that 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 applied, 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.

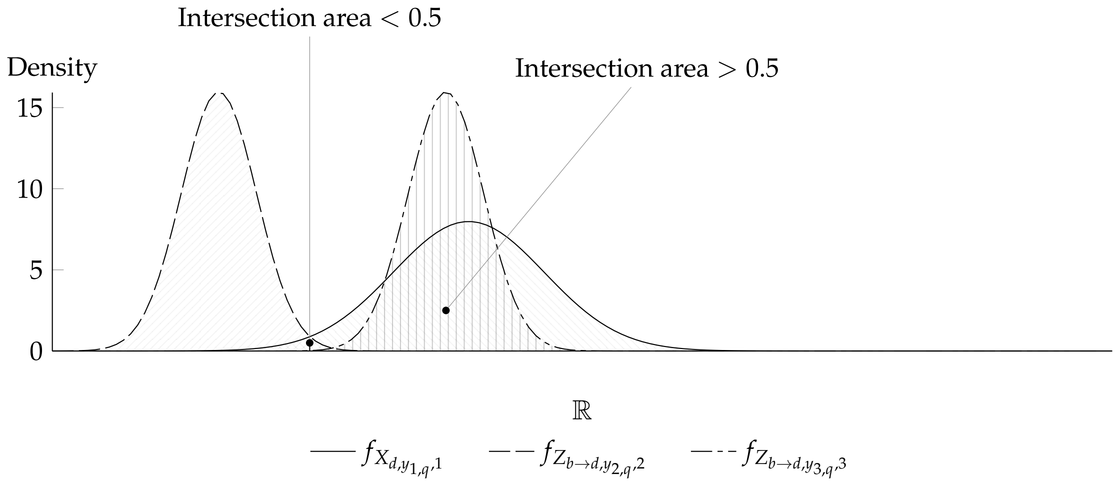

Figure 4. Hypothetical examples of the PDFs $f_{X_{d,y_{1,q}},1}$, $f_{X_{b,y_{2,q}},2}$ and $f_{X_{b,y_{3,q}},3}$, after applying the DA function $z_{b \to d}$.

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 a increase in the mean and the variance of the PDFs. In other words, if the PDF $f_{X_{d,y_{i,q}}}$ follows the normal distribution $\mathcal{N}(\mu, \sigma^2)$, then $f_{X_{b,y_{i,q}}}$ follows the normal distribution $\mathcal{N}(\texttt{const}_1 + \mu, \texttt{const}_2 + \sigma^2)$.

As a result, in order for the DA transformation function, $z_{b \to d}$, to undo the effect of the domain variation, by representing samples in the domain b in the domain d of the learning set, it adjusts the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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.

4. 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 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.

4.1. Vocabulary Richness

For any text $x_{i,d}$, vocabulary richness measures [12] aim to quantify the vocabulary diversity of input text $x_{i,d}$ for the purpose of solving stylometry problems. Examples of such measures are:

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

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

  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 to a word, a number, a punctuation mark, etc.
- 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 length of text $x_{i,d}$). To minimize this, Yule's K [16] and Honore's R [17] are functions that aim to stabilize the measures, as defined below:

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

where K_i is Yule's K measure for text $x_{i,d}$, and N_{words_w} is 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 $x_{i,d}$.

4.1.1. 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.

4.1.2. 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, while texts of varying target classification labels generally tend to maintain sufficiently different richness values.

However, such assumption is often false as the richness methods are found to be heavily and systematically dependent on the length of input texts, and that methods Yule's K and Honore's R that

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

attempt to stabilize them have questionable results [12]. More recently, [18] shows that Yule's K is ineffective for identifying authors.

4.2. Classical n-Grams

Classical n-grams (or just n-grams as commonly referred to in the literature) are probably the most common data representation method that is 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: n, gram, and, the length of the input text $x_{i,d}$, $\mathrm{len}(x_{i,d})$, which are defined as follows:

a) 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 and punctuations). 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 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*", "*for*", etc, ignoring any other types of words. In this case, $x_{i,d}[g]$ denotes the g^{th} function word in text $x_{i,d}$.

  Function words are traditionally identified by linguists on per language basis[2]. 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

[1] Examples of word separators are: paragraph start, punctuation marks, and whitespace.
[2] A list of function words in the English language can be found here: http://www.sequencepublishing.com/academic.html

in a given corpus (i.e. the most frequent words in a corpus are likely to mostly contain function words).

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.

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.

Example grams: $x_{i,d}[1] = \text{DT}$, $x_{i,d}[2] = \text{JJ}$, $x_{i,d}[3] = \text{NN}$, ..., $x_{i,d}[8] = \text{NN}$, where DT, JJ and NN are POS tags of corresponding words as tagged by Stanford's sentence parser[3]. The tags are defined as per the Penn treebank[4].

- 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.

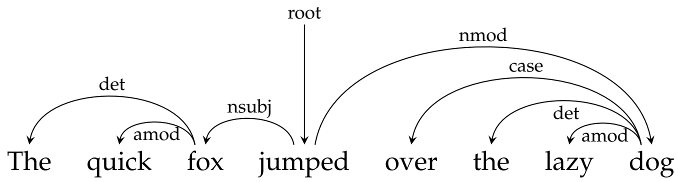

Figure 6. Dependency-based parse tree for the sentence "*the quick fox jumped over the lazy dog*" as identified by Stanford's statistical sentence parser.

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 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".

[3] http://nlp.stanford.edu:8080/parser/index.jsp
[4] https://www.cis.upenn.edu/~treebank/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

- Compound grams: theoretically a gram could be defined as a tuple of multiple grams. To the best of our knowledge, the concept of compound grams is novel, and has not been explored in the literature yet. The promising aspect of such compound grams is their ability in capturing the joint distribution of parts of texts taking certain 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`.

b) n: this parameter defines the width of the sliding window in the unit of grams, which is described in Algorithm 1 in this section. For example, if $n = 3$ and grams are *words*, then the width of the sliding window is 3 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.

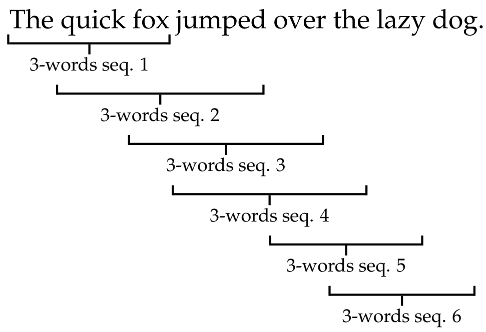

Figure 7. n-grams sliding window for an example sentence where $n = 3$ and grams are *words*.

c) $\text{len}(x_{i,d})$: this 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) that are found by any classical n-gram implementation is given by Algorithm 1, where $ng_i[j]$ denotes the j^{th} sequence of grams that is found by the searching algorithm from the input text $x_{i,d}$.

Algorithm 1 Pattern search by classical n-grams

for all $j \in \{1, 2, \ldots, \text{len}(x_{i,d}) - n + 1\}$ **do**
$\quad ng_i[j] = (x_{i,d}[j], x_{i,d}[j+1], x_{i,d}[j+2], \ldots, x_{i,d}[j+n-1])$
end for

For 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: $ng_i[1] = (\text{the}, \text{quick}, \text{fox})$, $ng_i[2] = (\text{quick}, \text{fox}, \text{jumped})$, $ng_i[3] = (\text{fox}, \text{jumped}, \text{over})$, $ng_i[4] = (\text{jumped}, \text{over}, \text{the})$, $ng_i[5] = (\text{over}, \text{the}, \text{lazy})$, and $ng_i[6] = (\text{the}, \text{lazy}, \text{dog})$.

4.2.1. 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 $x_{i,d}$. In order to facilitate meaningful comparison between representations of different text, the j^{th} component has to consistently

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

refer to the frequency of a specific n-gram. 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 in order to preserve space and reduce noise.

4.2.2. 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-gram implementations can be applied on any language.

On the other hand, n-grams assume that only patterns that are made of adjacent grams are patterns that are helpful. This is usually true (specially with natural languages), however it is not necessarily always true. This is often a limitation that is found in n-grams as they are able to only find those patterns that are made of adjacent n-grams. To address this issue, other data representations are proposed in the literature, some of which are variations of n-grams.

Therefore, augmenting n-grams by other feature extraction methods can potentially lead to the identification of more patterns that are useful for the classification tasks at hand.

4.3. k-Skip n-Grams

k-skip n-grams aim at generalizing classical n-grams such that grams within an n-gram sequence 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 sequence as presented in Algorithm 2.

Algorithm 2 Pattern search by k-skip n-grams

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

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: $kng_i[1] = (\text{the}, \text{quick}, \text{fox})$, $kng_i[2] = (\text{the}, \text{fox}, \text{jumped})$, $kng_i[3] = (\text{the}, \text{jumped}, \text{over})$, $\ldots$, $kng_i[25] = (\text{over}, \text{the}, \text{lazy})$, $kng_i[26] = (\text{over}, \text{lazy}, \text{dog})$, $kng_i[27] = (\text{over}, \text{the}, \text{dog})$, and $kng_i[28] = (\text{the}, \text{lazy}, \text{dog})$.

Note that the skips are only used to add an amount of tolerance (up to k skips) with regards to the grams adjacency within a grams sequence; 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 n-gram sequences.

4.3.1. Relation to Our Notation

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

4.3.2. 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:

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

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

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

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

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

4.4. Syntactic n-Grams

Classical n-grams identify n-gram sequences based on their order of appearance in their source texts. I.e. n-gram sequences 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-gram sequences are no longer needed to be spatially adjacent in their source texts, but rather adjacent in the syntactic tree representation of their source texts instead.

For example, in order 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* [19]. Figure 6 presents such dependency-based parse tree for the example sentence.
2. Then, identify n-gram sequences such that the identified grams are adjacent in the parsed tree. This can often lead to identifying n-gram sequences 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: $sng_i[1] = (\text{jumped}, \text{fox})$, $sng_i[2] = (\text{fox}, \text{quick})$, $sng_i[3] = (\text{fox}, \text{the})$, $sng_i[4] = (\text{jumped}, \text{dog})$, $sng_i[5] = (\text{dog}, \text{lazy})$, $sng_i[6] = (\text{dog}, \text{the})$, and $sng_i[7] = (\text{dog}, \text{over})$.

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

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 [19] 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: $sng_i[1] = (\text{S}, \text{NP}, \text{DT})$, $sng_i[2] = (\text{NP}, \text{DT}, \text{The})$, $sng_i[3] = (\text{NP}, \text{JJ}, \text{quick})$, $sng_i[4] = (\text{NP}, \text{NN}, \text{fox})$, $sng_i[5] = (\text{S}, \text{NP}, \text{JJ})$, $sng_i[6] = (\text{S}, \text{NP}, \text{NN})$, $sng_i[7] = (\text{S}, \text{VP}, \text{VBD})$, $sng_i[8] = (\text{VP}, \text{VBD}, \text{jumped})$, etc.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

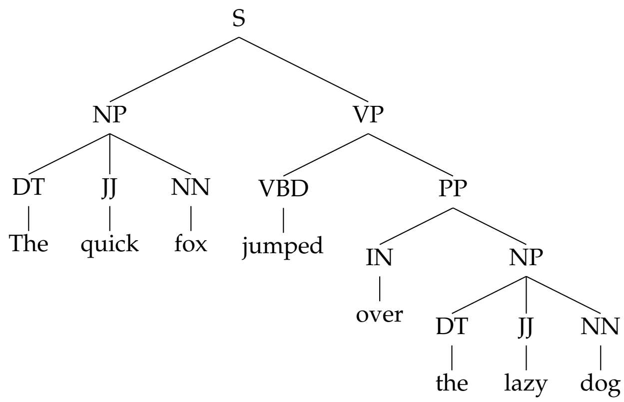

Figure 8. Constituency-based parse tree for the sentence "*the quick fox jumped over the lazy dog*" as identified by Stanford's statistical sentence parser.

4.4.1. Relation to Our Notation

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

4.4.2. Discussion

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

However, for the purpose of stylometry analysis, their disadvantages are that, unlike classical and k-skip n-grams, syntactic n-grams might not identify some gram sequences that are spatially adjacent. This is due to the fact that syntactic n-grams strictly walk over syntactic trees which can possibly result in missing some potentially important (for the purpose of 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.

Furthermore, depending on the implementation of the parser, while syntactic n-grams can be asymptotically computationally more scalable than k-skip n-grams (as the former might not have the combinatorial explosion problem that the latter has), current parser implementations are still computationally significantly more expensive than classical n-grams, as well as k-skip n-grams for when k is sufficiently small. Therefore, while syntactic n-grams can be asymptotically scalable, their dependency on sentence parses render them to be generally slow specially for stylometry problems that involve large volumes of text.

4.5. A Generalization of n-Gram-based Methods

All n-gram-based features 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 *spatial*, while in syntactic n-grams the direction is either *dependency-based tree* or *constituency-based tree*. For brevity, we will refer to these direction as `spatial`, `deptree` and `constree`, respectively.
- l specifies the minimum number of times a given sequence of grams must occur. For example, if $l = 5$ then only those sequences that occur for 5 times or greater will be used to represent text samples.
- k, n and grams are as defined in previous sections.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Further details about this generalization is presented in Section 5 about our extensive feature extraction library, Fextractor.

4.6. Rewrite Rules

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

To the best of our knowledge, the use of rewrite rules in the literature of electronic text stylometry is so far restricted to Context-Free Grammars (CFGs) that are 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 rewrite rules are found: S $\rightarrow$ NP + VP, NP $\rightarrow$ DT + JJ + NN, DT $\rightarrow$ The, JJ $\rightarrow$ quick, NN $\rightarrow$ fox, etc.

In order to reduce the amount of irrelvant information that is made available by the terminal nodes (e.g. content words), rewrite rules that lead to terminal nodes could be removed.

Alternatively, (while unseen in the literature) it is possible to substitute such terminal nodes by other values in order 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 be: S $\rightarrow$ NP + VP, NP $\rightarrow$ DT + JJ + NN, DT $\rightarrow$ Ccc, JJ $\rightarrow$ ccccc, NN $\rightarrow$ ccc, etc.

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

4.6.1. 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 that is uniquely identified by j.

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

4.6.2. Discussion

The advantages of rewrite rules as features is that they can capture syntactic structures in text $x_{i,d}$. Specially 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, function words, etc).

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

4.7. Raw Text

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

Such stylometry methods are largely inspired by the work of Tomàš Mikolov et al. on the construction of language models via Recurrent Neural Networks (RNNs) [20]. A notable example of a stylometry problem solver that analyzes raw text is the work of Douglas Bagnall [21], where raw texts were analyzed to construct language models, using RNNs, on per-author basis with the objective 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, certain

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

information were removed from the input texts at a pre-processing stage (e.g. replacing all numbers by a placeholder).

4.7.1. 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 $x_{i,d}$.

4.7.2. Discussion

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

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

However, the disadvantage is that excess information can negatively affect the run-time and space requirements of the solver, as well as potentially confuse the learning algorithm or over-fit the training samples. To avoid confusing the learning algorithm or over-fitting the training samples, a pre-processing stage might be necessary to remove the excess information.

5. Fextractor: Extensive Stylometry Feature Extraction Library

One of the key issues that face today's research on stylometry is the fact that implementations of most of the stylometry-related proposed methods are not released publicly. As a result, re-evaluating, or comparing newer methods against the previous ones is often extremely difficult due to the need of re-implementing those methods again (which requires a tremendous amount of time and effort).

A notable aspect of the research in electronic text stylometry, is the enhancement of feature extraction methods. Currently, such methods are highly diverse, and range from simple The letter counts, up to more sophisticated ones that use independent statistical models, such as POS taggers. For example, it is quite common in the literature that that a good portion of the proposed feature extraction methods are evaluated in isolation, without adequate comparison against existing methods to truly justify their relative effectiveness. Another issue is the lack of adequate generalizations of the proposed methods, which leaves some of the novel variants unstudied.

Our contributions that are presented in this section are:

- The generalization of numerous feature extraction methods. This allows us to define novel variants of the existing feature extraction methods, in addition to simplifying the implementation.
- The implementation of an extensive stylometry feature extraction library with easy-to-use interface in Python. While alternative feature extraction libraries exist [22], to the best of our knowledge, our library Fextractor is, by far, the most extensive library of its kind to date. Our library supports language-independent features, as well as language-dependent features for languages Arabic, Chinese, French, German, and Spanish.
- The release of the library under a permissible open-source library. We hope that this would enable other researchers to conveniently study the feature extraction methods, or evaluate their methods against the existing ones, without facing the time and effort barrier that is currently required

 doi:10.20944/preprints201708.0003.v1

to implement the many methods. This can be found in the Git repository https://gitlab.com/mmaakh/fextractor.git.

5.1. Supported Feature Extraction Methods

The following feature extraction methods are supported:

- n-grams (classical n-grams), with parameters:
  - Normalize (boolean): If set to `True`, the library will normalize the total quantity of the n-gram sequences by sum of all frequencies of n-grams of the same kind. If set to `False`, then raw frequency counts are returned.
  - l: The minimum frequency an n-gram sequence must have in order for the library to list it. E.g. if $l \leq 1$, then all n-gram sequences are returned. If $l = 5$, then only those that occur for at least 5 times are returned.
  - n: The size of the sliding window of n-gram, in the unit of grams.
  - Gram: The definition of gram. Table 2 presents a list of supported grams.
  - Cache (path): If set to `None`, then caching is disabled. If set to a path, then caching is enabled. This can be useful for expensive features, such as those that require making use of POS taggers (the cache will save time by avoiding parsing same sentences twice).
- k-skip n-grams [23], with parameters:
  - k: the total number of tolerate adjacency violations in an n-gram sequence, in the unit of grams. E.g. $k = 2$ will tolerate up to 2 adjacency violations, while $k = 0$ will not tolerate any and cause it to be identical to classical n-grams.
  - Normalize (boolean).
  - l.
  - n.
  - Gram.
  - Cache (path).
- Syntactic n-grams when (using dependency trees) [19], with parameters:
  - Normalize (boolean).
  - l.
  - n.
  - Gram.
  - Cache (path).
- Rewrite-rules. Unlike other rewrite-rules implementations, ours has the novelty in that it allows us to substitute the terminal words by their alternative forms (e.g. word shape, word shape, etc). For consistency, we refer to this as "gram". Additionally, compound grams are also made available to the rewrite-rules feature extraction function. The parameters are:
  - Normalize (boolean).
  - l.
  - Gram. Table 2 lists all supported gram definitions.
  - Cache (path).
- Richness measures (Hapax legomena, unique words rate).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 2. The currently supported definitions of grams as used by the at least l-frequent `dir`-directed k-skip n-grams, and rewrite-rules feature extraction methods.

Gram	Compound	Generalized n-grams	Rewrite rules	Languages
`letter`	No	Yes	No	Any
`word`	No	Yes	Yes	Any
`wordlen`	No	Yes	Yes	Any
`wordshape`	No	Yes	Yes	Any with A-Z
`wordshape-word`	Yes	Yes	Yes	Any with A-Z
`funcword`	No	Yes	Yes	English
`pos`	No	Yes	Yes	Arabic, Chinese, French, German, and Spanish
`dep`	No	Yes	No	Arabic, Chinese, French, German, and Spanish
`word-pos`	Yes	Yes	No	Arabic, Chinese, French, German, and Spanish
`word-dep`	Yes	Yes	No	Arabic, Chinese, French, German, and Spanish
`pos-dep`	Yes	Yes	No	Arabic, Chinese, French, German, and Spanish

5.2. Generalization of n-Gram Methods

This section presents the mechanism by which our library implements n-grams, k-skip n-grams, and syntactic n-grams. In order to simplify the implementation, enhance the ability of introducing more novel variants, as well as extending the coverage of the library, we have generalized all of the n-gram-based methods as the at least l-frequent `dir`-directed k-skip n-grams. Then, we implemented this generalization instead. As a result of this, we get a more-extensive library that is also simpler and allows superior code re-use. Further details are presented below.

Consider the text example that is presented in Figure 6, and, for simplicity, suppose that grams are defined to be *words*. Then, if the parameter `dir` = `spatial`, the example text in Figure 6 is represented in the following row matrix in (16).

$$\begin{bmatrix} \text{The} & \text{quick} & \text{fox} & \text{jumped} & \text{over} & \text{the} & \text{lazy} & \text{dog} \end{bmatrix} \tag{16}$$

Then, the sliding window, as depicted in Figure 7, will operate on the matrix in (16) on row-by-row basis. Since there is a single (but long) row, the sliding window will move along that one row, depending on the chosen value of parameter n.

On the other hand, if the parameter `dir` = `deptree`, the example text in Figure 6 is represented in the following row matrix in (17). Note that each row represents a path from the `root` node towards the numerous leaf nodes as we walk down the dependency tree that is depicted in the Figure.

$$\begin{bmatrix} \text{jumped} & \text{fox} & \text{quick} \\ \text{jumped} & \text{fox} & \text{The} \\ \text{jumped} & \text{dog} & \text{over} \\ \text{jumped} & \text{dog} & \text{the} \\ \text{jumped} & \text{dog} & \text{lazy} \end{bmatrix} \tag{17}$$

Then, similar to the `dir` = `spacial` case, the sliding window, as depicted in Figure 7, will operate on the matrix in (17) on row-by-row basis. Since there 5 rows, the sliding window will move along

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

each row, independently. It can be seen that, because of this design, we are able to re-use our sliding window code for both `dir` = `spacial` (classical n-grams) and `dir` = `deptree` (syntactic n-grams with dependency trees).

Alternatively, one may decide to construct a matrix similar to the one in (16), but while using a different type of trees, or methods that might not necessarily be based upon linguistics basis. Extending this library is as simple as introducing code that defines a matrix out of sentences.

For completeness, (18) and (19) present variants of the matrices (16) and (17), respectively, except for defining grams to be POS tags.

$$\begin{bmatrix} \text{DT} & \text{JJ} & \text{NN} & \text{VBD} & \text{IN} & \text{DT} & \text{JJ} & \text{NN} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \text{VBD} & \text{NN} & \text{JJ} \\ \text{VBD} & \text{NN} & \text{DT} \\ \text{VBD} & \text{NN} & \text{IN} \\ \text{VBD} & \text{NN} & \text{DT} \\ \text{VBD} & \text{NN} & \text{JJ} \end{bmatrix} \tag{19}$$

As for the parameter k that specifies the total number of permissible gram skips, it is implemented in the sliding window code, and is therefore fully re-used elsewhere, independent of the direction. Therefore, the rest of the code is re-used, independent of how the matrices are defined.

5.3. Examples

The example below demonstrates how to obtain the unique words richness measures for the example in the Figure 6, which prints 0.875.

```python
# load the library
import fextractor

# define some text
text = 'the quick fox jumped over the lazy dog'

# compute a richness score
score = fextractor.get_richness_unique(text)

# print the score
print(score)
```

The example below demonstrates how to represent the same example sentence based on the raw frequency of its CFG rules. Note that, in order to parse the sentences, the address of a Stanford CoreNLP[5] HTTP server should be specified. If the server address is not specified, the it defaults to the URL http://corenlp.run/.

```python
# load the library
import fextractor

# define some text
text = 'the quick fox jumped over the lazy dog'

# find the cfg rewrite rules
r = fextractor.getcount_rewriterules(text, 'en', gram='word', normalize=False,
                                     server='http://corenlp.run')

# print the score
print(r)
```

Which prints the rewrite-rules:

```
{"[u'DT', [u'the']]":            2,
 "[u'IN', [u'over']]":           1,
```

[5] https://stanfordnlp.github.io/CoreNLP/

```
 "[u'JJ', [u'lazy']]":               1,
 "[u'JJ', [u'quick']]":              1,
 "[u'NN', [u'dog']]":                1,
 "[u'NN', [u'fox']]":                1,
 "[u'NP', [u'DT', u'JJ', u'NN']]":   2,
 "[u'PP', [u'IN', u'NP']]":          1,
 "[u'ROOT', [u'S']]":                1,
 "[u'S', [u'NP', u'VP']]":           1,
 "[u'VBD', [u'jumped']]":            1,
 "[u'VP', [u'VBD', u'PP']]":         1}
```

If `gram='wordlen'`, then terminal nodes are replaced by their lengths as follows:

```
{"[u'DT', ['3']]":                   2,
 "[u'IN', ['4']]":                   1,
 "[u'JJ', ['4']]":                   1,
 "[u'JJ', ['5']]":                   1,
 "[u'NN', ['3']]":                   2,
 "[u'NP', [u'DT', u'JJ', u'NN']]":   2,
 "[u'PP', [u'IN', u'NP']]":          1,
 "[u'ROOT', [u'S']]":                1,
 "[u'S', [u'NP', u'VP']]":           1,
 "[u'VBD', ['6']]":                  1,
 "[u'VP', [u'VBD', u'PP']]":         1}
```

The example below demonstrates how to represent the same example sentence as a matrix based on syntactic n-grams.

```
# load the library
import fextractor

# define some text
text = 'the quick fox jumped over the lazy dog'

# find the syntactic n-gram matrix
m = fextractor.get_grams(text, 'en', gram='pos', direction='deptree')

# print the score
print(m)
```

Which give the output (note how the output resembles the matrix in 18):

```
[[u'VBD', u'NN', u'DT'],
 [u'VBD', u'NN', u'JJ'],
 [u'VBD', u'NN', u'DT'],
 [u'VBD', u'NN', u'IN'],
 [u'VBD', u'NN', u'JJ']]
```

The matrix can then be used in order to identify sequences of grams. Below is an example for identifying all 2-skip 2-grams, along with their normalized frequencies.

```
# load the library
import fextractor

# define some text
text = 'the quick fox jumped over the lazy dog'

# find the syntactic n-gram matrix
m = fextractor.get_grams(text, 'en', gram='pos', direction='deptree')

# count raw k-skip n-grams patterns with raw frequencies
p = fextractor.getcount_ksngrams(m, k=2, n=2, normalize=False)

# print the score
print(p)
```

Which prints the output:

```
{u'NN::DT':   2,
 u'NN::IN':   1,
 u'NN::JJ':   2,
 u'VBD::DT':  2,
 u'VBD::IN':  1,
 u'VBD::JJ':  2,
 u'VBD::NN':  5}
```

 doi:10.20944/preprints201708.0003.v1

If `normalize=True`, then the output would be normalized as follows:

```
{u'NN::DT':  0.13,
 u'NN::IN':  0.06,
 u'NN::JJ':  0.13,
 u'VBD::DT': 0.13,
 u'VBD::IN': 0.06,
 u'VBD::JJ': 0.13,
 u'VBD::NN': 0.33}
```

Additionally, if using a vector-representation is required, the represented texts can be trivially converted into vectors. Below is a code of an example where two distinct texts, `text1` and `text2`, are transformed into a vector space.

```
# load the library
import fextractor

# define some text
text1 = 'the quick fox jumped over the lazy dog'
text2 = 'this is an example of a different text'

# find the syntactic n-gram matrix
m1 = fextractor.get_grams(text1, 'en', gram='pos', direction='deptree')
m2 = fextractor.get_grams(text2, 'en', gram='pos', direction='deptree')

# count raw k-skip n-grams patterns with raw frequencies
p1 = fextractor.getcount_ksngrams(m1, k=2, n=2, normalize=True)
p2 = fextractor.getcount_ksngrams(m2, k=2, n=2, normalize=True)

# agree on some order for the components
pmaster = {i for i in list(p1) + list(p2)}
features_order = list(pmaster)

# represent them as vectors
x1 = []
x2 = []
for f in features_order:
    if f in p1:
        x1.append(p1[f])
    else:
        x1.append(0)

    if f in p2:
        x2.append(p2[f])
    else:
        x2.append(0)

# print the vectors
print('x1 = ' + str(x1))
print('x2 = ' + str(x2))
```

Which will print the vector-representation of the texts as follows:

```
x1 = [0.13, 0.13, 0.06, 0.00, 0.13, 0.13, 0.00, 0.33, 0.06]
x2 = [0.00, 0.16, 0.16, 0.08, 0.33, 0.00, 0.25, 0.00, 0.00]
```

Such vectors could then be used by other classifiers as required.

6. Stylometry Problems

The problems can be defined as follows:

- Solvers of AA and AP problems are special cases of SCC if no domain variation among 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, $\mathcal{Y}_q = \{10s, 20s, \ldots\}$.
- Solvers of AV problems [9] are special cases of SOC if no domain variation exists among the analyzed samples, and MOC if otherwise. The targeted labels set $\mathcal{Y}_q$ is defined as the set of author identities.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

- 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 due to the fact that such problems can be de-composed 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).

6.1. Key Challenges

BigData problem scenarios make interesting domains for stylometry analysis. However, as stylometry methods were mostly executed against controlled datasets, they do not scale very well when executed against BigData scenarios, such when genre or topic variations exist among the analyzed documents [24,25].

The most notable stylometry scalability challenges are faced when:

- The size of the suspect authors set is too large.
- The size of the analyzed documents is too small.
- Or when the analyzed documents belong to varying topics, genres or times (i.e. the domain variation problem).

Some of these scalability issues are being tackled recently, such as cross-topic AA [26] by which cross-topic texts per author is evaluated to be helpful for enhancing the accuracy of AA solvers, and AA on small messages [27].

Additionally, a key challenge that faces the implementation of stylometry problem solvers in forensics application domains is the fact that stylometry problem solvers are known to be vulnerable to adversarial attacks. Brennan et al. [28] evaluate that non-linguistic texts from authors that:

- Manually obfuscate their writing style will degrade the classification accuracy of evaluated AA solvers down to that of random chance guessing.
- Manually imitate the writing style of some target victim will degrade the classification accuracy of the evaluated AA solvers below that of random chance guessing.

Following this line of research, Khonji et al. [29]:

- Independently re-evaluated findings of [28] by different AA solvers, and confirm that they are statistically significant.
- Extended the evaluation in [28] by also evaluating AV solvers and conclude that the same findings also apply (despite the fact that AA and AV problems are special cases of different fundamental problems SCC and SOC, respectively).
- Conjectured that the findings should also apply to other stylometry problem solvers, namely AC, AP, and AD. This conjecture is based on the fact that such problems are also special cases of close-set (SCC, MCC) and open-set (SOC, MOC) problems.

6.1.1. Discussion

We believe that obfuscation and imitation attacks can be thought of as special cases of the domain variation problem, where imitation and obfuscation are distinct domains. E.g. adversaries change the domains of their writings by obfuscating their writing styles, or imitating other victims.

Therefore, we conjecture that supervised and unsupervised domain adaptation methods can be promising tools in enhancing the performance of MCC and MOC stylometry problem solvers, respectively, against adversarial attacks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

7. Stylometry Problem Solvers

7.1. General-purpose Learning Algorithms

Support Vector Machines (SVMs) are generally regarded among the most accurate general-purpose learning algorithms for solving SCC problems [12]. Examples of use of such models in the literature are [30–34].

Decision trees are commonly used to solve SCC problems [35–38]. However, recently decision trees were also used to solve AV problems (a special case of SOC problems) [39].

Similarly, Artificial Neural Networks (ANNs) are used to solve SCC problems [37,40–43]. Recently, Bagnall [21] successfully used RNN to solve AV problems by constructing language models by analyzing input texts, ultimately allowing to estimate the probability of having a given input testing text to be written by a given author.

Other SCC estimation methods include discriminant analysis [44–46], memory-based [47], and probabilistic methods [48–51].

7.2. Common n-grams

Keselj, et al. proposed an estimation of SCC in [52] 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, in relation to 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:
$$\text{cng}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \sum_{c \in \{1,2,\dots\}} \left(\frac{2(\mathbf{x}_{i,d}[c] - \mathbf{x}_{j,d}[c])}{\mathbf{x}_{i,d}[c] + \mathbf{x}_{j,d}[c]} \right)^2$$
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 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 $\mathcal{X}_L$.

An advantage of Common n-grams (CNG) is its independence on the language of learning and testing text samples. This makes CNG also applicable for programming languages. A significantly simplified variant of CNG is proposed in [53] 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 $\mathbf{x}_{j,d}$.

More recently, an ensemble of cng is proposed in [54] to solve the AV problem (a special case of SOC problems) by which two input samples $\mathbf{x}_{i,d}$ and $\mathbf{x}_{j,d}$ are considered to be written by the same author if their CNG score $\text{cng}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})$ is below certain threshold value.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

7.3. Compression

For any samples $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}$ Keogh et al. propose in [55] the following similarity measurement function:

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

where comp : $\mathcal{X} \to [0, \infty)$ is a function that returns total number of bits after compressing its input using some compression algorithm (e.g. GZIP, RAR, etc) 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 [56] as follows:

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

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:

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

or by substituting the cdm function by ncd.

Graaff et al. [57] and Veenman et al. [58] evaluated cdm as estimations of solvers of SCC and SOC. However, the classification accuracy of such methods is less accurate than the state of the art methods. We believe that this is due to the fact that compression methods aim to classify based on the total entropy of input text samples, which can also include irrelevant information relating to the samples domain as opposed to the target classification task at hand.

7.4. Burrows Delta

Burrows delta [59] and its variants are amongst 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 $\text{zscore}(\mathbf{x}_{i,d}[c]) = \frac{\mathbf{x}_{i,d}[c] - \mu_c}{\sigma_c}$ 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:

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

Variations of Burrows delta essentially substitute the Manhattan distance by other distance measures. For example, the quadratic delta Δ_Q [60]:

$$\Delta_Q(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \sum_{c \in \{1,2,\dots\}} \left(\text{zscore}(\mathbf{x}_{i,d}[c]) - \text{zscore}(\mathbf{x}_{j,d}[c]) \right)^2$$

and the cosine delta $\Delta_\angle$:

$$\Delta_\angle(\mathbf{x}_{i,d}, \mathbf{x}_{j,d}) = \frac{\sum_{c \in \{1,2,\dots\}} \mathbf{x}_{i,d}[c] \mathbf{x}_{j,d}[c]}{\sqrt{\sum_{c \in \{1,2,\dots\}} \mathbf{x}_{i,d}[c]} \sqrt{\sum_{c \in \{1,2,\dots\}} \mathbf{x}_{j,d}[c]}}$$

Jennidis et al. empirically evaluate variations of Burrows delta measure and find that the cosine delta $\Delta_\angle$ is the most accurate measure amongst the evaluated variants [61]. Evert et al. further analyze the findings in [61] and show that $\Delta_\angle$'s higher accuracy is due to the effect of normalizating

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

vectors, and that the other variants (e.g. Δ_B) can be as accurate as $\Delta_\angle$ if the vectors are normalized (i.e. $||\mathbf{x}_{i,d}|| = ||\mathbf{x}_{i,d}|| = 1$, where $||\mathbf{x}_{i,d}||$ denotes the magnitude of the vector $\mathbf{x}_{i,d}$).

7.5. Unmasking

Koppel et al. propose the unmasking algorithm as an estimated solver to the AV problem [9]. For any $\mathbf{x}_{i,d}, \mathbf{x}_{j,d} \in \mathcal{X}_T$, the unmasking algorithm aims to answer the question 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 that are written by same authors are harder to separate (or classify as different authors) than texts that are written by different authors. More specifically, the unmasking algorithm solves the AV problem as follows:

1. As stated in Section 6, 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 texts 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 in order to allow for the text parts to be large enough for subsequent analysis [62].
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 with the task of predicting 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 are plotted as a function of each feature removal step, a classification *degradation curve* is found.
4. If the degradation curve is sufficiently steep, then it is assumed that $y_{i,q} = y_{j,q}$, otherwise $y_{i,q} \neq y_{j,q}$ is assumed.

Alternatively, instead of following the iterative approach to construct the degradation curve as outlined earlier, Koppel et al. [9] propose a computationally simpler variation that only evaluates the joint PDF of number of features and their Information Gain (IG) with respect to the same classification task (classifying input text parts in $\mathbf{x}_{i,d}$ and $\mathbf{x}_{j,d}$ with the assumption that they belong to different clusters). If the density quickly drops as a function of increasing the IG, then $y_{i,q} = y_{j,q}$ is assumed, otherwise $y_{i,q} \neq y_{j,q}$ is assumed.

An evaluation in [9] shows that the unmasking algorithm is also an effective estimation of the solver of the multi-topic AV problem (i.e. a special case of the MOC problem). An independent evaluation by Luyckx et al. [25] show that while the unmasking algorithm is effective at single-genre AV problems ($f_1 = 0.6198$; the f_1 score is the harmonic mean of precision and recall), multi-genre AV problems remain significantly more difficult to solve ($f_1 = 0.5572$).

7.6. Impostors

Koppel et al. [63] propose the *impostors* algorithm as an estimation of 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 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 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:

$$s \leftarrow s + \begin{cases} \frac{1}{r} & \text{if } \begin{pmatrix} \mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})^2 > \\ \mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{m}_{i,d}) \times \mathrm{sim}(\mathbf{x}_{j,d}, \mathbf{m}_{j,d}) \end{pmatrix} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

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 been also successfully evaluated in the literature, such as the following as adopted by Khonji et al. [64]:

$$s \leftarrow s + \frac{\mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{x}_{j,d})^2}{\mathrm{sim}(\mathbf{x}_{i,d}, \mathbf{m}_{i,d}) \times \mathrm{sim}(\mathbf{x}_{j,d}, \mathbf{m}_{j,d})} \tag{21}$$

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.

Worth noting that both of the Impostors-based AV solvers, that of Koppel et al. [63] and that of Khonji et al. [64], ranked as the most accurate competing AV solvers in the author identification competitions PAN'13 [13] and PAN'14 [14], respectively.

8. Features Evaluation Methodology

In Section 4.5, we generalized many features 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 permitting the identification of previously-unevaluated features.

One of the key gaps that exists in the current state of the literature 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 due to the fact that the evaluations follow non-unified testing beds (e.g. different testing datasets, different evaluation methodology).

Another issue is that most of the existing evaluations derive conclusions without performing statistical significance tests. Therefore it is unknown, even within their isolated testing beds, 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 unaccessible to the community. This results in slowing down the pace of research as different research groups would need to re-implement the methods.

To address the issues above, 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 re-usability of our developed tools by releasing all our associated evaluated datasets and developed tools openly under a permissible

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

open source license. Additionally, our tools are implemented by a friendly programming language (e.g. Python).

The following subsections will outline the specifics of the points above.

8.1. 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 `dir` $\in$ {`spatial`, `deptree`}.
- 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.

8.2. Evaluation Problems

The evaluation problems follow the SCC scenario. Recall that SCC stylomemtry 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'12 author identification competition, which is the latest PAN competition on SCC author identification problems. Since this is an author identification scenario, 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 [65]. This dataset is composed of a set of IMDb reviews as authored by some of the "prolific" users of IMDb [65]. Originally, this dataset is composed of 62,000 text files in total. However, we reduced the dataset down to 1,000 files by 8 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 due to the fact that our evaluation 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.

Table 3. Datasets statistics.

Dataset name	S24	S1000
Dataset source	PAN'12 Prob. C [66]	IMDb [65]
#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

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) are fed into a Random Forest (RF) classification learning algorithm. The output of this RF learning algorithm is an SCC classification model.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

- The remaining one third of the text files are used as testing samples. Target classification labels of these testing are predicted by the RF model that is trained earlier. The prediction output of this RF model are logged for future analysis as detailed in Section 8.3.

In order 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 that exists 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 $l \in \{2, 3, \ldots, 99\}$. 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.

8.3. 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 distinct 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.

In order 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:

$$\text{acc}_i = \frac{\sum_{o \in O_i} \begin{cases} 1 & \text{if prediction } o \text{ is correct} \\ 0 & \text{otherwise} \end{cases}}{\text{total number of problems}} \tag{22}$$

- The statistical significance of the difference in the measured accuracies 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 is due to random noise. This is also referred to as the *null hypothesis*.
  - H_1: the difference between acc_i and acc_j is 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 differences 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 accuracies, namely $|\,\text{acc}_i - \text{acc}_j\,|$, or greater absolute differences, can arise under the hypothesis H_0. We refer to this probability as the p value.

In order 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 [14] as presented in Table 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 4. Statistical significance levels.

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

8.4. Evaluation Reproducibility

The feature extraction Python module, namely `fextractor.py`, 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 code to generate evaluation model outputs, and *visualization* contains code to translate the model outputs into the figures and tables that are presented in Section 9. The dependencies are Python[6], Scikit-learn[7], CoreNLP[8], and Matplotlib[9].

9. Features Evaluation Results

The objective of this evaluation is to identify properties of the feature extraction functions that correspond to increase in classification accuracy. Since this evaluation tests many feature extraction functions that are special cases of the at least l-frequent `dir`-directed k-skipped n-grams, the properties that we evaluate their effects on the classification accuracy are l, `dir`, k, n, and grams.

Additionally, since the at least l-frequent `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.

9.1. Independent Parameters Evaluation

As discussed in Sections 8.1 and 8.2, 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

---

6 https://python.org/
7 http://scikit-learn.org/
8 https://stanfordnlp.github.io/CoreNLP/
9 http://matplotlib.org/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

the evaluation datasets, the RFs algorithm is used to construct AA classification models, which 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 entire results in a single figure. However, due to significant challenges in representing items in 4, or greater, dimensional spaces, this subsection will present the accuracy measurements independently for each parameter. 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, if the parameter is the definition of gram, then 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 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 values of the parameter allows for the existence of the highest classification accuracy values. We find the use of ECDFs curves more appealing than PDFs in this evaluation, since plotting them requires minimal assumptions about the distribution of the accuracy measurements at hand. This is different than when PDFs are plotted, which, depending on the method, requires explicit definitions of the bandwidth and the kernel, as is the case with Kernel Density Estimation (KDE).

9.1.1. Parameter: l

Figure 9 presents the classification accuracy of each of the 23, 760 RF classification models on dataset S24, from the perspective of the parameter l. More specifically, the figure presents 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 represent the accuracy of $23,760/99 = 240$ RF many classification models.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

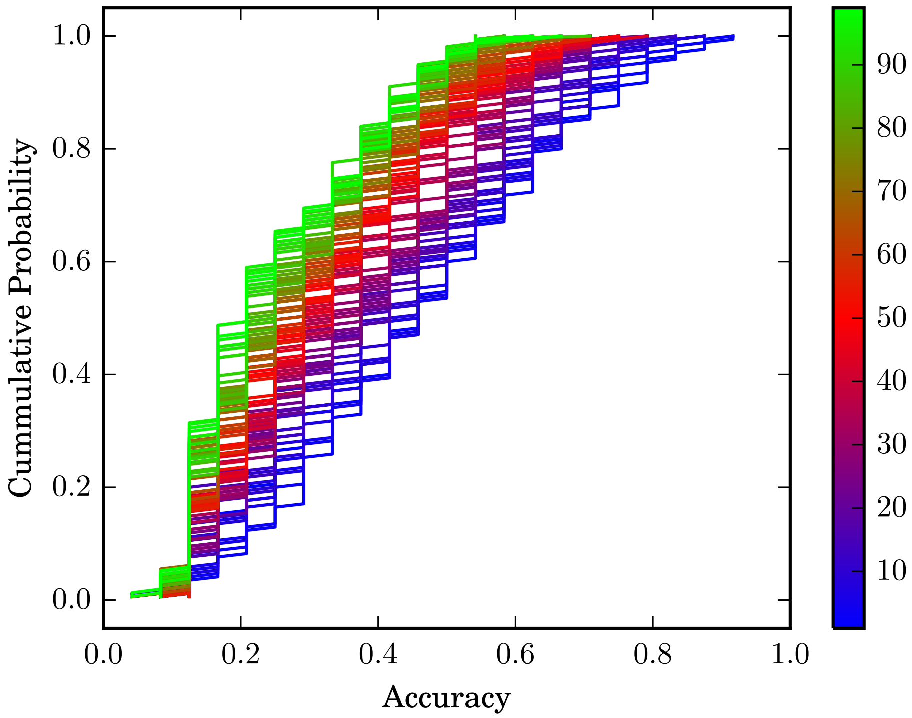

Figure 9. The ECDFs of classification accuracy from the perspective of the parameter l (for all $l \in \{1, 2, \ldots, 99\}$ against the S24 problems set).

Similarly 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 due to the fact that 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).

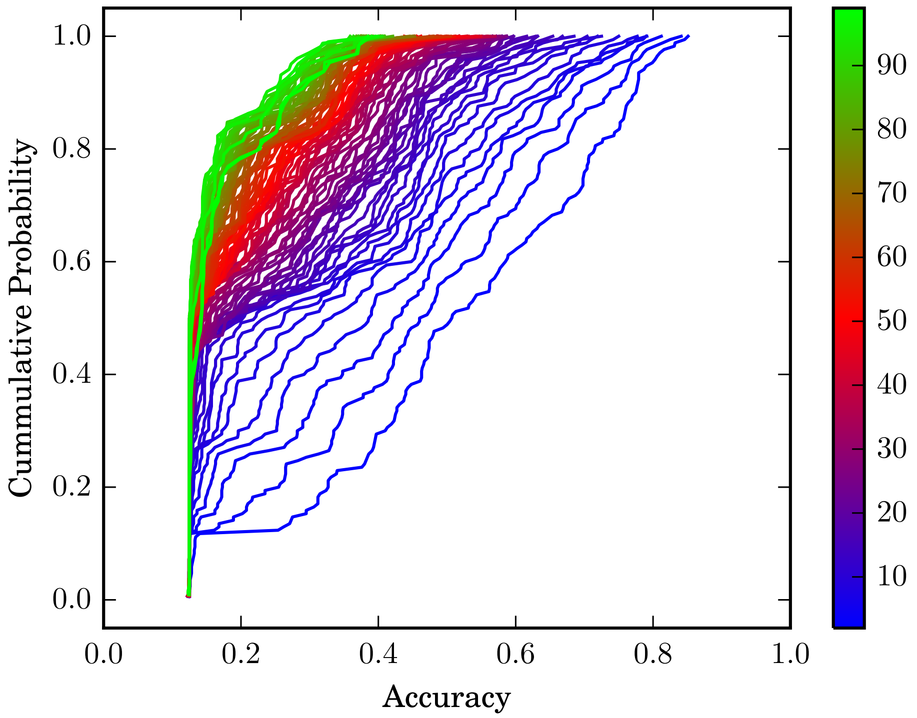

Figure 10. The ECDFs of classification accuracy from the perspective of the parameter l (for all $l \in \{1, 2, \ldots, 99\}$ against the S1000 problems set).

1205 We can see from the ECDFs in Figures 9 and 10 that the evaluations on both of the datasets, S24
1206 and S1000, agree in that the selection of lower values of l can allow for the identification of more
1207 accurate classification models.
1208 Specifically, Tables 5 and 6 list the highest classification accuracy for each $l \in \{1, 2, \ldots, 4\}$, along
1209 with their corresponding pair-wise p values (note that $l = 1$ is not evaluated on the dataset S1000 for
1210 the reason given in Section 8.2).

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	–	$p = 0.0020$ **

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

It can be seen from the 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 is able to show that the increase in the classification accuracy with $l \in \{2, 3\}$, relative to $l = 4$, is statistically highly significant ($p < 0.001$).

However, it is important 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 is able to identify and use robust features, while ignoring harmful (or useless) features.

Beyond learner's ability in identifying useful features from harmful ones comes the datasets ability in presenting adequate information to the learner. For example, if there exists domains mismatch across learning and testing samples, it can be impossible for a learning algorithm to set useful feature apart from the harmful ones. Therefore, in such situations, one may use his domain knowledge to identify properties of the useful or harmful features. In the case when low frequent features correspond mostly to harmful features, one may increase the value of l in order to eliminate most of the harmful features.

Therefore, the fact that the classification models in our experiment have managed to maximize their classification accuracy by reducing the value of the parameter l is possibly an indication of the robustness of RF against noisy features, and that the evaluation datasets are adequately controlled such that at least no domains mismatch exists between samples that share the same target classification label. However, we see no implications of this observation with respect to domains mismatch across instances of *different* target classification labels.

9.1.2. Parameter: `dir`

When the parameter $n = 1$, the `dir` parameter is irrelevant. This is due to the fact that `dir` determines the direction by which multiple grams are identified to form a single n-gram sequence. When $n = 1$, such sequences cannot be formed, and therefore the parameter `dir` becomes irrelevant. 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 of the Figures 11 and 12, on datasets S24 and S1000, agree in that higher classification accuracy levels can be achieved when $\texttt{dir} = \texttt{spatial}$ than when $\texttt{dir} = \texttt{deptree}$. 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 the Figures 11 and 12 disagree on the effectiveness of $\texttt{dir} = \text{N/A}$. Figure 11 suggests that $\texttt{dir} = \texttt{spatial}$ can allow for the identification classifiers that are more accurate than those that can be identified by the case when $\texttt{dir} = \text{N/A}$, while Figure 12 suggests 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

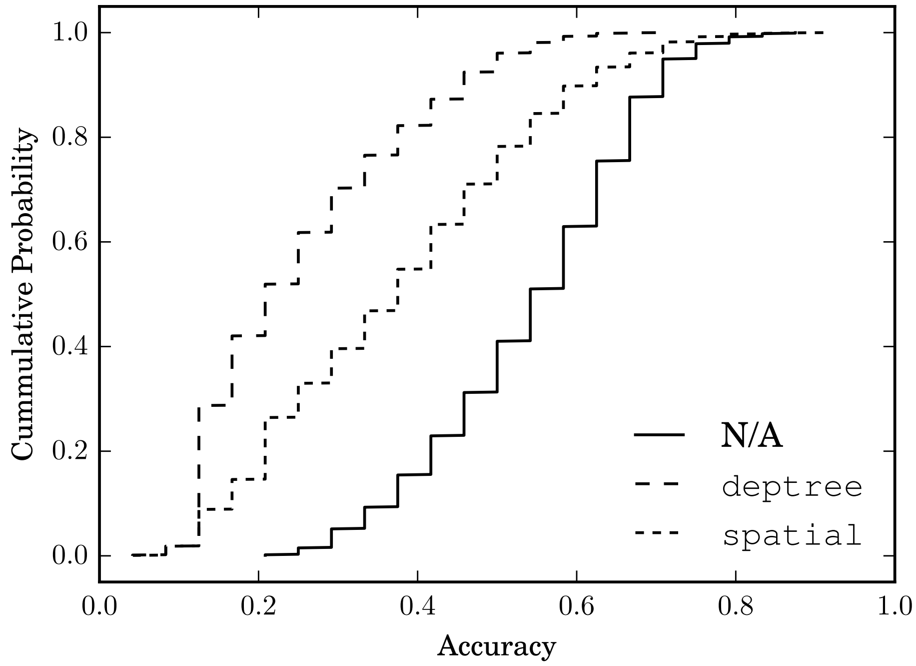

Figure 11. The ECDFs of classification accuracy from the perspective of the sliding window movement direction of n-grams (S24 problems set).

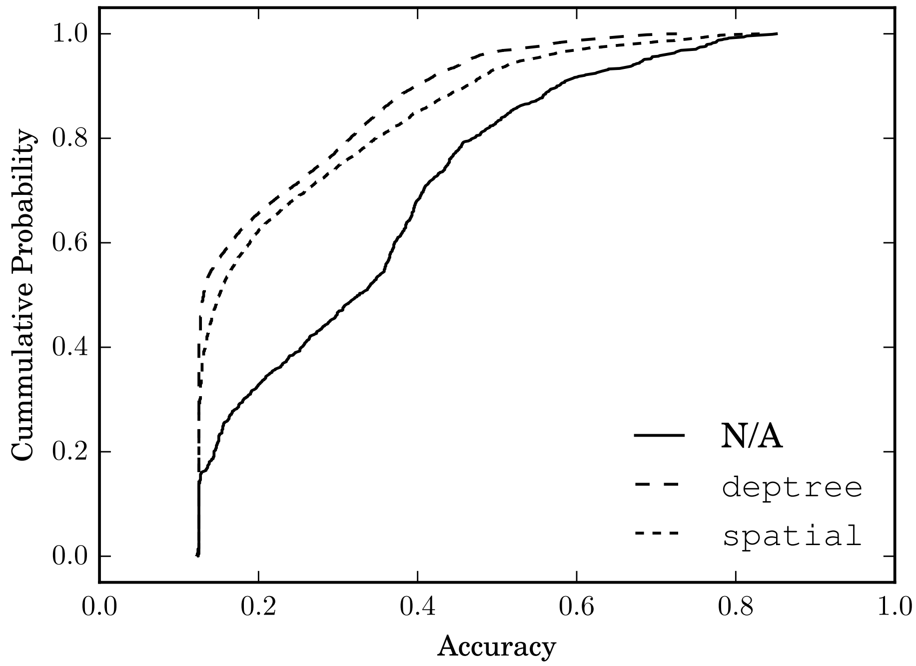

Figure 12. The ECDFs of classification accuracy from the perspective of the sliding window movement direction of n-grams (S1000 problems set).

 doi:10.20944/preprints201708.0003.v1

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 $= 0.73$	–	$p = 0.0009$ ***

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 with respect to the parameter `dir` to attempt to answer some of the question of Section 9.

We know that n-grams and k-skip n-grams follow the `spatial` direction, while syntactic n-grams with dependency trees follow the `deptree` direction. Therefore, having both of the datasets S24 and S1000 agree on 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 with respect to their ability in identifying features that result in higher classification accuracies.

We also find this to be consistent with our intuition as dependency trees carry a considerable amount of information with respect to the semantics (or the content) that are embodied within the analyzed texts. This can cause learning algorithms to over fit the topic of the analyzed texts as opposed to their writing style. In other words, it causes the author identification models to be partly topic identification models.

However, it remains unclear whether this superiority of `spatial` is because of factors that are specific to n-grams, or whether it is because of factors that are specific to k-skip. We will answer this question in Section 9.2.

9.1.3. 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, 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 accuracies 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 ($p = 0.0549$).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

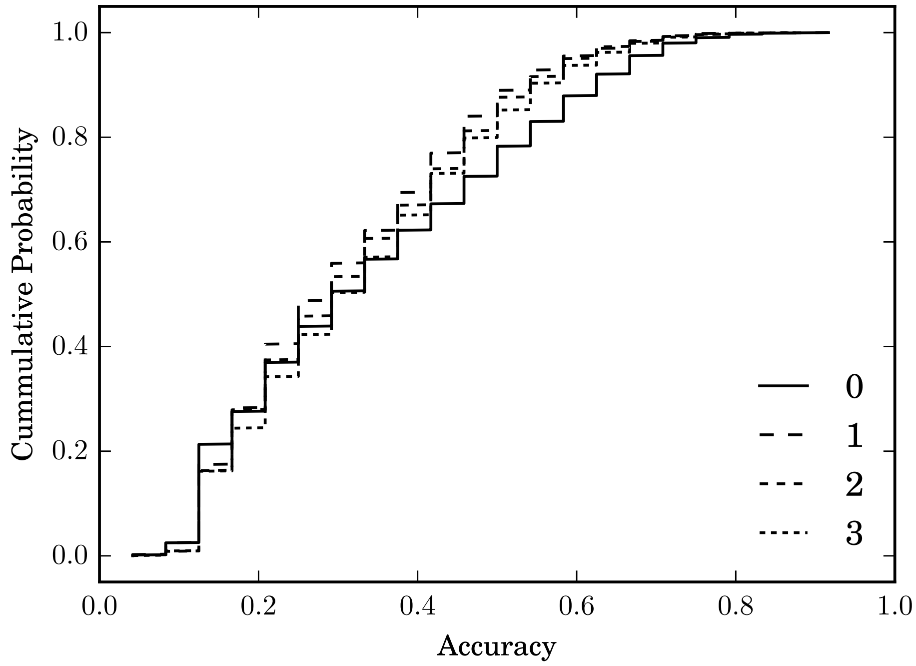

Figure 13. The ECDFs of classification accuracy from the perspective of the parameter *k* (S24 problems set).

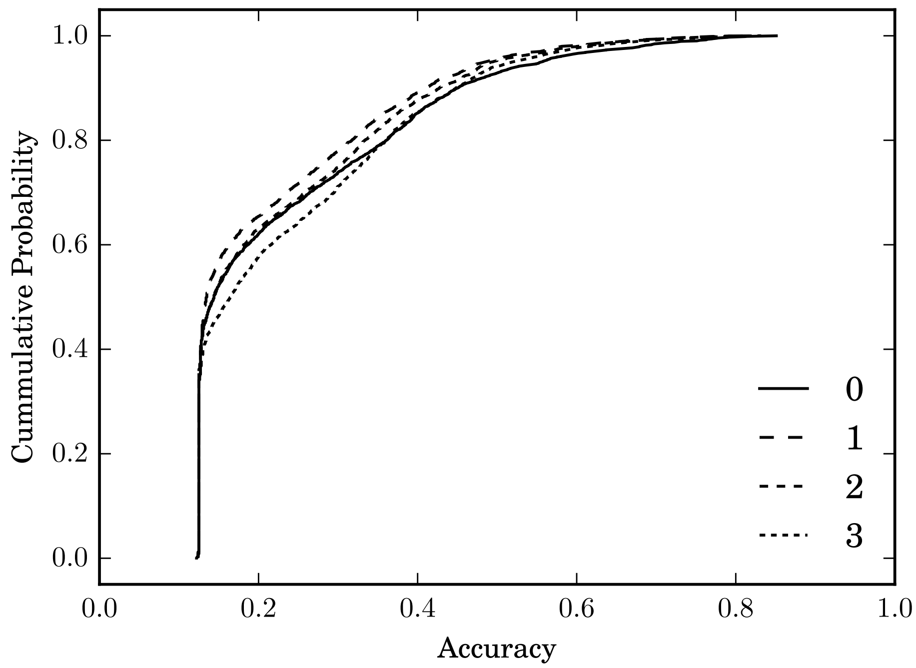

Figure 14. The ECDFs of classification accuracy from the perspective of the parameter *k* (S1000 problems set).

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 $= 0.83$	–	–	$p = 0.8961$ $=$

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

9.1.4. Parameter: n

Figures 15 and 16 suggest that both of the datasets S24 and S1000 agree in that bi-grams (i.e. $n = 2$) can result in the identification of more accurate classification models than can 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, the 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 suggests the opposite while also showing statistical significance on dataset S1000.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

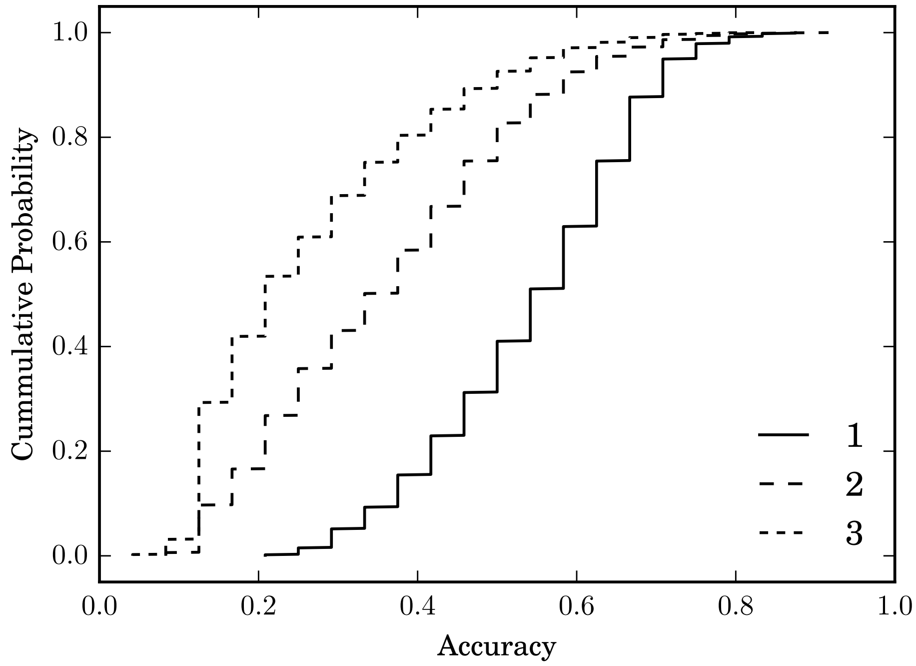

Figure 15. The ECDFs of classification accuracy from the perspective of the parameter n (S24 problems set).

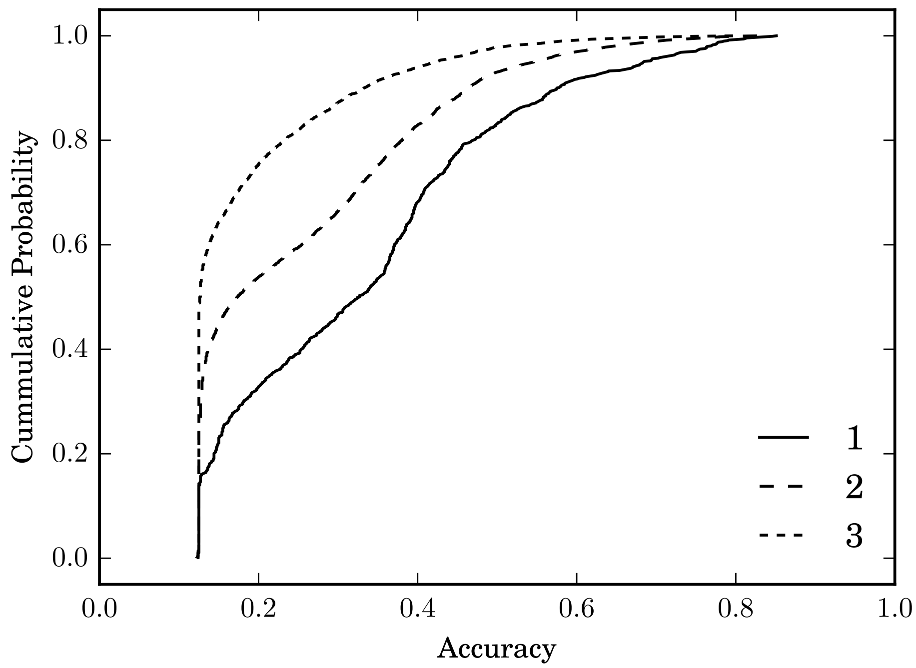

Figure 16. The ECDFs of classification accuracy from the perspective of the parameter n (S1000 problems set).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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 $= 0.92$	–	$p = 0.3147$ =

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 $= 0.83$	–	$p = 0.0030$ **

9.1.5. Parameter: gram

Figures 17 and 18 agree in that the gram that has the lowest upper bound limit on its accuracy 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 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) is unable to demonstrate statistical significance of any of the differences between the grams. The only exception where dataset S24 can show a statistically significant difference is with respect to `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`. In fact, both of the grams have lead 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` ($p = 0.0619$).

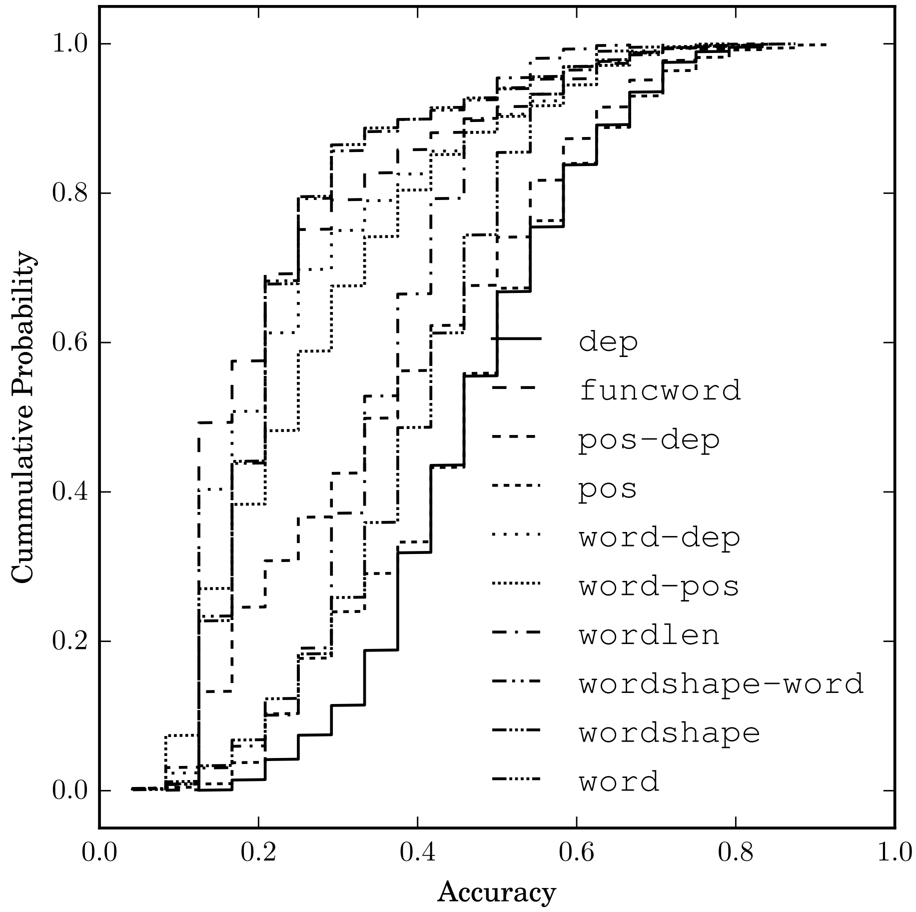

Figure 17. The ECDFs of classification accuracy from the perspective of grams (S24 problems set).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

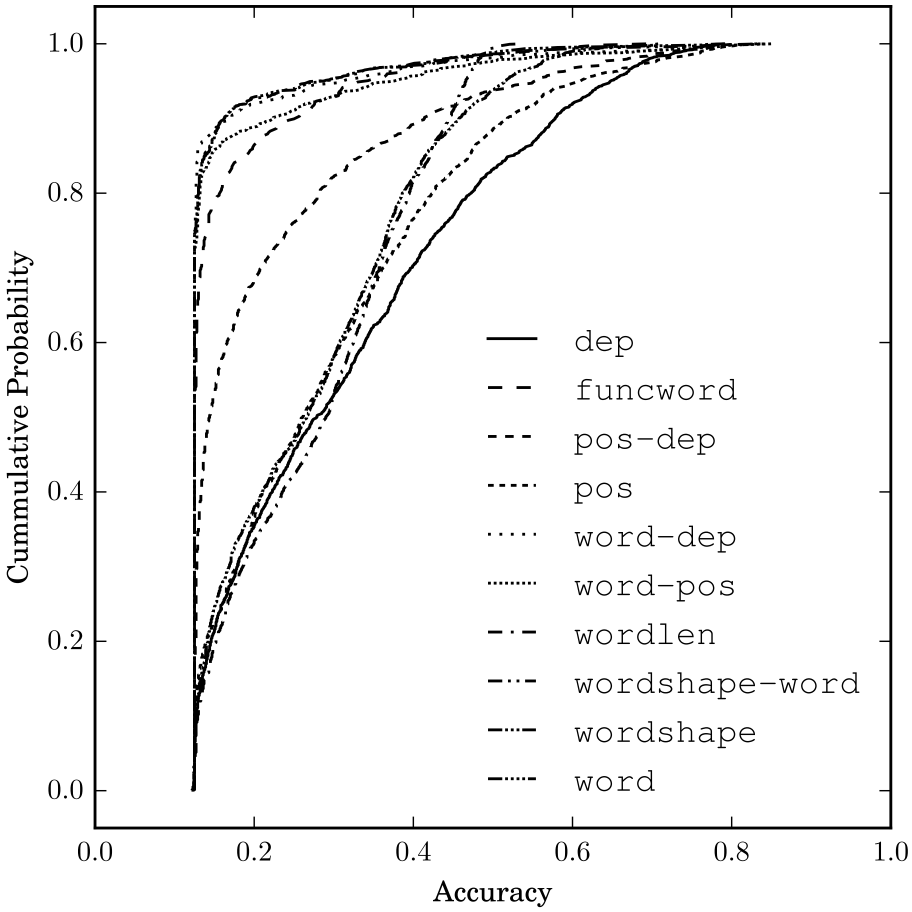

Figure 18. The ECDFs of classification accuracy from the perspective of grams (S1000 problems set).

9.2. Dependent Parameters Evaluation

Section 9.1 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, in order to answer some of the questions in Section 9. Specifically, with respect to the directions `spatial` and `deptree`.

In order 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 tied on achieving the maximum classification are identified.
2. Then, the list of the most accurate feature extraction methods is compared against all other feature extraction methods in order to identify other feature extraction methods that are similar enough to the most accurate methods. In order 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 in order to identify the cluster of the least accurate feature extraction methods, by which the pair-wise statistical significance tests are performed against the least accurate feature extraction methods (instead of the most accurate methods).

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

- 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 for the purpose of solving 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 values of l suggest that they relatively identify reliable patterns that help the learning algorithm (RF) to identify accurate classification models. However, this observation may differ depending on the noise that is 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 fairly diverse, and not dominated by $k = 0$. This implies that k-skip n-grams as features are indeed a step forward in identifying classification patterns that allow for more accurate stylometry problem solvers. Similarly, values of the parameter n is fairly diverse, which suggests the usefulness 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, has a slight edge. However, this is only shown to be statistically significant in the dataset S1000.

By observing the list of the most 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 9.1 that larger values of l generally correspond to lower classification accuracy).

In order 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 that follow 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 for the identification of 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.

10. Evaluating Author Attribution on Emirati Tweets

While various stylometry problem solvers are evaluated against texts of various domains, the performance of AA methods remain unknown against electronic texts of Emirati social media. The contributions of this section are:

- The construction of the first AA evaluation dataset based on Emirati tweets (the KIT-30 dataset).
- The novel definition of grams, which we denote by *compound grams*, that allow for achieving significantly higher classification accuracies than possible with classical definitions.
- The first evaluation of AA classification models against such Emirati tweets.

The most related evaluation to this study is that of the PAN'12 closed-set AA challenge [66], where a number AA models are evaluated against several problems, including closed-set AA ones. While all of the closed-set AA PAN'12 datasets fall under the English language, this evaluation remains significant as it presents the classification accuracy of the state of the art in solving AA problems.

Khonji et al. have shown in [38] that, when representing texts by the frequency of their characters, function words, the at least l-frequent n-grams, rewrite-rules, word lengths, word shapes, etc, then the RFs classification algorithm can identify classification models that achieve classification accuracies that are identical to those of the most accurate AA models of the closed-set AA evaluation of PAN'12. The only exception is with respect to the *Problem C* of PAN'12, where the RFs model misclassified only a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

single problem more than the best performing AA model. However, this single misclassification is not statistically significant.

Other author identifications evaluation problems in the literature, including the subsequent iterations of PAN competitions, diversified their pool of languages in their evaluation datasets. E.g. the following datasets were used in recent PAN evaluation: Dutch, Greek and Spanish. However, evaluation of stylometry methods, such as AA solvers, against Emirati texts remained absent in literature.

The most related evaluation dataset to our constructed KIT-30 is perhaps the Arabic Sentiment Tweets Dataset (ASTD) by Nabil et al. [67]. However, while our methods of obtaining the tweets are inspired by those of Nabil et al., the ASTD dataset has the authors' identifiers discarded (rightfully so due to the nature of Twitter terms of use, as well as the nature of the study that the ASTD dataset was constructed for). Therefore, ASTD is not usable for the purpose of evaluating AA models.

10.1. The Khonji-Iraqi Emirati Tweets Author Identification Evaluation Dataset (KIT-30)

This section introduces the objective of our dataset, methods that were used in order to construct it, as well as its various statistics.

The objective of the KIT-30 dataset is to offer texts that are suitable for creating and solving Emirati author identification problems, for the purpose of evaluating author identification models. Recall that an author identification could be the AA problem, but could also be the AV problem (verifying whether a pair of texts is written by one author, or two distinct authors), or the AD problem (clustering, say, paragraphs in a single document by their authors).

In order to achieve the objectives above, we follow the steps in Algorithm 3. This resulted in obtaining a total number of 30 Emiraty Twitter accounts.

Algorithm 3 Obtaining a set of Twitter user accounts.

1. Identify a number of the most active Twitter accounts in the UAE by using *SocialBakers*. The use of SocialBakers is inspired by Nabil et al. [67].
2. Identify more Twitter accounts by searching for certain tags that are often local to the UAE. This step is also inspired by Nabil et al. [67].
3. Manually inspect all of the the Twitter accounts, and discard those that do not seem to originate from UAE, or those that do not tweet in Arabic.

Then, Algorithm 4 was used to download, discard, and pre-process the tweets as deemed appropriate for the objectives of the evaluation dataset at hand. This resulted in obtaining the finalized KIT-30 dataset, which is comprised of over $50,000$ tweets in total.

Algorithm 4 Downloading and preprocessing tweets.

1. For any Twitter account, download as many tweets as permissible by the Twitter API.
2. All re-tweets are discarded. This is due to the fact that re-tweets contain texts that are written by authors other than those of the account owners.
3. All tags, user names, and URLs are replaced by place holders. This is in order to ensure that the evaluated author identification models remain unable solve author identification problems by simply memorizing specific tags, user names, or URLs that potentially happen to strongly correlate with their author identities. For example, all hashtags, such as "#الإمارات", become the uniform placeholder hashtag "#TAG".
4. Unlike the ASTD dataset, we store author identifiers. This is necessary for the dataset in order to be usable for creating and solving author identification problems.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1400 In order to facilitate meaningful comparisons between the evaluation results against this Emirati
1401 tweets dataset and those of other languages, we have repeated the same process by adapting it for the
1402 Dutch, Greek, Spanish, and English languages.
1403 The overall statistics of KIT-30 is presented in Table 13 and Figure 19. The per-author statistics of
1404 Emirati tweets are presented in Table 14. The per-author statistics of the other languages are presented
1405 in Appendix B.

Table 13. Overall KIT-30 statistics.

Dataset	#Authors	#Tweets
Emirati tweets	30	51957
Dutch tweets	30	62018
Greek tweets	30	63622
Spanish tweets	30	59996
US tweets	30	70837

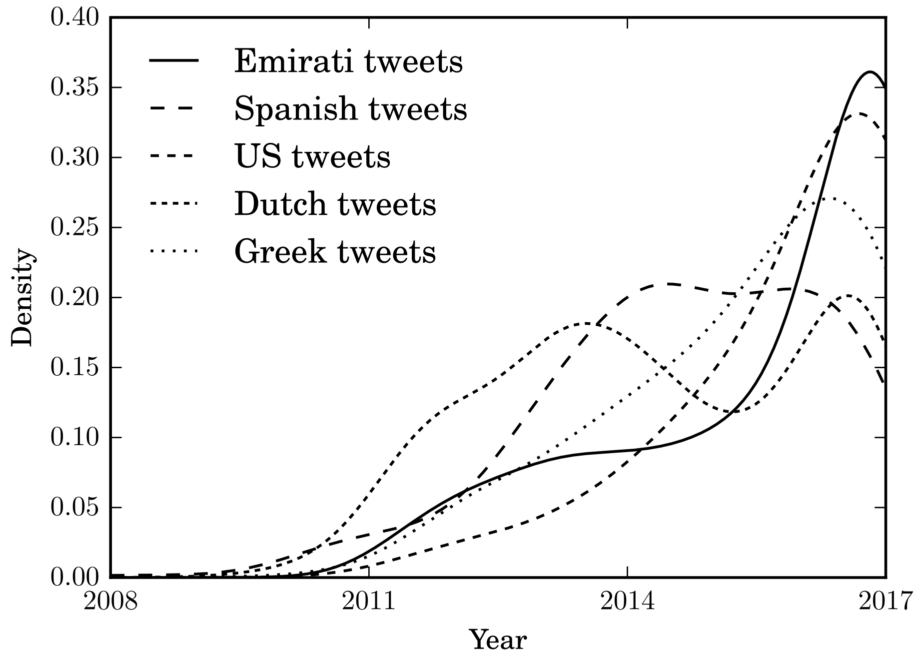

Figure 19. Estimated PDF of the distribution of tweet times by language.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 14. KIT-30 statistics of the Emirati tweets subset.

Author ID	#Tweets	Avg. #letters/tweet	Avg. #words/tweet
AE-1	2457	90.5332	13.5173
AE-2	2067	104.5568	17.5288
AE-3	2500	72.6344	12.7564
AE-4	612	61.4428	10.4575
AE-5	1760	88.5733	15.0494
AE-6	820	64.8451	11.7390
AE-7	1229	80.3539	14.2555
AE-8	2890	34.1789	6.5481
AE-9	2255	91.7282	16.3623
AE-10	720	95.0931	16.5736
AE-11	1734	118.3339	20.7814
AE-12	1153	101.8829	16.8049
AE-13	3010	62.2884	11.7492
AE-14	1464	105.6243	17.9481
AE-15	2034	57.7670	10.0565
AE-16	3043	52.5008	9.4512
AE-17	1652	55.5884	9.2615
AE-18	711	81.7032	14.9044
AE-19	2496	53.9291	10.1046
AE-20	830	58.5614	10.5819
AE-21	2226	74.6168	12.2255
AE-22	69	104.5072	17.2754
AE-23	578	74.7803	12.9291
AE-24	2510	71.9845	12.2175
AE-25	1810	67.2713	11.5199
AE-26	1034	84.5551	14.6122
AE-27	1435	104.9624	18.0042
AE-28	1422	79.9951	13.5598
AE-29	2765	97.2976	17.0304
AE-30	2671	85.6050	16.3669

The implementation of the Tweets downloader in Step 1 of Algorithm 4, along with the finalized KIT-30 dataset are released under permissible open-source licenses[10].

10.2. Author Attribution Model

We adopt RFs as the learning algorithm to find the AA classification model for solving AA problems in the domain of Emirati social media electronic texts. This is due to the fact that, as shown in [38], the algorithm achieves competitive high classification accuracy when solving AA problems.

RFs require the learning and testing samples to be represented in a vector space. We also follow a vector representation method that is inspired by that of Khonji et al. [38]. In other words, for any text x, x is represented as a dim dimensional vector $\mathbf{x}$, such that for any component $i \in \{1, 2, \ldots, \mathrm{dim}\}$, $\mathbf{x}[i]$ represents the frequency of a unique k-skip n-gram pattern [23] (or sequence) in the text x.

In order to facilitate meaningful comparison between any pair of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ that represent different texts x_1 and x_2, respectively, component i always refers to the frequency of the same unique k-skip n-gram pattern, except that $\mathbf{x}_1[i]$ is the frequency of this pattern in text x_1, while $\mathbf{x}_2[i]$ is the frequency of the same pattern in text x_2.

Then an RF model is trained by vector-represented texts of the learning set along with their author identities. The resultant model is then used to predict the author of unseen texts against the space of suspects whose texts are represented in the learning set.

[10] http://khonji.org/stylometry

 doi:10.20944/preprints201708.0003.v1

Before we define k-skip n-gram patterns, we define n-gram patterns, and then expand the definition of n-grams by the addition of k-skips.

An n-gram pattern is essentially a sequence of n many adjacent grams in a given text. For example, if the text at hand is "*the quick fox jumped over the lazy dog*", the grams are defined to be *words*, and $n = 3$, then all n-gram sequences of the example text are depicted in Figure 7.

The only novelty that k-skip n-grams bring relative to n-grams is that they expand each n-gram sequence into multiple sequences such that the grams adjacency constraint is allowed to be violated for up to k many skips.

For example, if $k = 2$, and the starting gram is "*The*", then we not only identify the 3-gram sequence "*The quick fox*", but also all of its 3-gram variants as listed in Table 15.

Table 15. k-skip n-grams in text "*The quick fox jumped over ...*" for when $k = 2$, $n = 3$, grams are defined to be words, and the first gram is "*The*". Struck-through grams denote skipped ones.

2-skip 3-words					Skips
The	quick	fox			0
The	~~quick~~	fox	jumped		1
The	quick	~~fox~~	jumped		1
The	~~quick~~	~~fox~~	jumped	over	2
The	quick	~~fox~~	~~jumped~~	over	2
The	~~quick~~	fox	~~jumped~~	over	2

Similar inflation occurs to all other n-gram sequences (except those near the end of the string by which only fewer skips become possible in order to avoid overrunning after the end of the string).

It can be seen that the concept of k-skip n-grams is a generalization of the concept of n-grams. This makes n-grams a special case of k-skip n-grams for when $k = 0$. I.e. 0-skip n-grams and n-grams are identical (for any value of n, and any definition of what constitutes as a gram).

Note that since k-skip n-grams inflate each n-gram sequence into multiple variants with skips ranging from 0 up to k (inclusive of 0 and k), the total number of n-gram parameters increase combinatorially. Therefore large values of k are sometimes computationally infeasible.

Additionally, it is common to ignore k-skip n-gram sequences that do not occur frequently enough. This is to reduce dimensionality, as well as due to the fact that such measures are often found to be too noisy for the purpose of solving AA problems. A successful rule that has been used in the literature is to ignore all k-skip n-gram sequences that only occur for less than l many times in any single text in the evaluation corpus. For example, our preliminary evaluations of the proposed models showed that $l = 5$ was optimum for the purpose of their evaluation (i.e. if a pattern fails to occur for 5 or more times in any text, it gets ignored and therefore not used in subsequent analysis).

10.3. Compound Grams

Table 15 presents 2-skip 3-grams, when grams are defined to be words. Another definition of grams that is known in the literature of stylometry is defining them to be the POS tags, or dependency tags. For example, Table 16 presents an example of the case when grams are defined to be POS tags. Note that each word is substituted by its corresponding POS tag, as defined by the Penn Treebank project[11]. The same could be trivially extended to dependency tags, word lengths, word shapes, etc.

[11] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

 doi:10.20944/preprints201708.0003.v1

Table 16. k-skip n-grams in text "*The quick fox jumped over ...*" for when $k = 2$, $n = 3$, grams are defined to be POS tags, and the first gram is the POS tag of "*The*". Struck-through grams denote skipped ones.

2-skip 3-words					Skips
DT	JJ	NN			0
DT	~~JJ~~	NN	VBD		1
DT	JJ	~~NN~~	VBD		1
DT	~~JJ~~	~~NN~~	VBD	IN	2
DT	JJ	~~NN~~	~~VBD~~	IN	2
DT	~~JJ~~	NN	~~VBD~~	IN	2

Compound grams essentially aim to aggregate multiple definitions of grams that refer to the same text segment. Table 17 presents examples of some compound grams, when aggregating the definition "*word*" and "*POS tag*" into one gram.

Table 17. k-skip n-grams in text "*The quick fox jumped over ...*" for when $k = 2$, $n = 3$, grams are defined to be tuple word-POS tags, and the first gram is the tuple word-POS tag that correspond to "*The*". Struck-through grams denote skipped ones.

2-skip 3-words					Skips
The-DT	quick-JJ	fox-NN			0
The-DT	~~quick-JJ~~	fox-NN	jumped-VBD		1
The-DT	quick-JJ	~~fox-NN~~	jumped-VBD		1
The-DT	~~quick-JJ~~	~~fox-NN~~	jumped-VBD	over-IN	2
The-DT	quick-JJ	~~fox-NN~~	~~jumped-VBD~~	over-IN	2
The-DT	~~quick-JJ~~	forx-NN	~~jumped-VBD~~	over-IN	2

Compound grams allow for capturing additional information than the classical ones. For example, measuring the frequencies of grams, as shown in Figures 15 and 16, allows for identifying the tendency of words or POS tags to independently occur in a given text. On the other hand, as shown in Table 17, measuring the frequency of compound grams allows for identifying the tendency of certain words to jointly take certain POS tags in a given text. This can be valuable information for identifying authors, as authors can be made unique not only by the independent frequency of certain grams (words or POS tags), but rather by their tendency of choosing certain words in certain positions of their sentences. For example, the word "*saw*" can be used as both, a verb, and a noun as in the sentence "*I saw the saw*".

10.4. Evaluation Methodology

Once AA models are trained as described in Section 10.2, we evaluate them by using 10-fold cross-validation. However, in order to ensure that each fold is comprised by realistic learning and testing samples, we add the constraint that limits the free mixing of tweets that were written at different times. Specifically, the tweets per author are chronologically grouped into 10 chunks such that their tweets do not exist in too adjacent time intervals.

This constraint increases the difficulty of the AA problems, as it essentially minimizes the possibility of test tweets of being chronologically too close from their learning counterparts.

The statistics of the evaluation dataset, after grouping the tweets into 10 chronological chunks on per author basis, is presented in Table 18. Those of the Dutch, Greek, Spanish, and US English languages are presented in Appendix B.

Table 18. Chunked KIT-30 statistics of the Emirati tweets subset.

Author ID	#Chunks	Avg. #letters/chunk	Avg. #words/chunk
AE-1	10	22244	3321.2
AE-2	10	21611.9	3623.2
AE-3	10	18158.6	3189.1
AE-4	10	3760.3	640
AE-5	10	15588.9	2648.7
AE-6	10	5317.3	962.6
AE-7	10	9875.5	1752
AE-8	10	9877.7	1892.4
AE-9	10	20684.7	3689.7
AE-10	10	6846.7	1193.3
AE-11	10	20519.1	3603.5
AE-12	10	11747.1	1937.6
AE-13	10	18748.8	3536.5
AE-14	10	15463.4	2627.6
AE-15	10	11749.8	2045.5
AE-16	10	15976	2876
AE-17	10	9183.2	1530
AE-18	10	5809.1	1059.7
AE-19	10	13460.7	2522.1
AE-20	10	4860.6	878.3
AE-21	10	16609.7	2721.4
AE-22	10	721.1	119.2
AE-23	10	4322.3	747.3
AE-24	10	18068.1	3066.6
AE-25	10	12176.1	2085.1
AE-26	10	8743	1510.9
AE-27	10	15062.1	2583.6
AE-28	10	11375.3	1928.2
AE-29	10	26902.8	4708.9
AE-30	10	22865.1	4371.6

Recall from earlier sections that the at least l-frequent k-skip n-grams have the following parameters:

- Parameter l.
- Parameter k.
- Parameter n.
- The definition of what constitutes a gram.

For completeness, we repeat the evaluation many times, each with a distinct AA RF model, such that each makes use of a unique data representation function. Specifically, we exhaustively implement all possible definitions of the at least l-frequent k-skip n-grams for the following sets of parameter values:

- $l \in \{1, 2, \ldots, 9\}$.
- $k \in \{0, 1\}$.
- $n \in \{1, 2, 3\}$.
- Gram $\in$ {word, word length, POS tag, word-POS tag tuple, dependency tag, word-dependency tag tuple, POS-dependency tags tuple}. The tuple grams essentially represent a special case of our proposed compound grams with two components.

This process results in $9 \times 2 \times 3 \times 7 = 378$ unique text vectorization methods, each of which is used by an RF model that is evaluated by a 10-fold cross-validation.

 doi:10.20944/preprints201708.0003.v1

The only exception to this is the Dutch and Greek datasets by which Gram $\in$ {word, word length}. This is due to the fact that the POS tagger that we use[12] does not support them.

Additionally, since the accuracy of AA problem solvers is sensitive to the size of the space of suspect authors, we repeat the entire evaluation for 29 times, each time while evaluating against a unique size of suspects space. I.e. we evaluate for all suspect space sizes in $\{2, 3, \ldots, 30\}$. Therefore, the total number of evaluations is $378 \times 29 = 10,962$ many 10-fold cross-validations.

Finally, Approximate Randomization (AR) is used in order to measure the p value for testing the statistical significance of various evaluation outcomes as appropriate. The labels of various significance levels are shown in Table 4.

11. Emirati Tweets Evaluation Results

11.1. Accuracy of Author Attribution Models as a Function of Suspects Space Size

Recall from earlier that, in total, 10,962 10-fold cross-validations are performed in order to evaluate RF AA models exhaustively with various parameters of the at least l-frequent k-skip n-grams.

Figure 20 presents the ECDF of all of the 10,962 classification accuracies that are obtained by 10-fold cross-validation against the Emirati tweets dataset, such that, for any line $i \in \{2, 3, \ldots, 30\}$ (each line i is denoted by a unique colour), i represents the ECDF of the classification accuracies of all models that are evaluated against problems with a suspects space of i many authors.

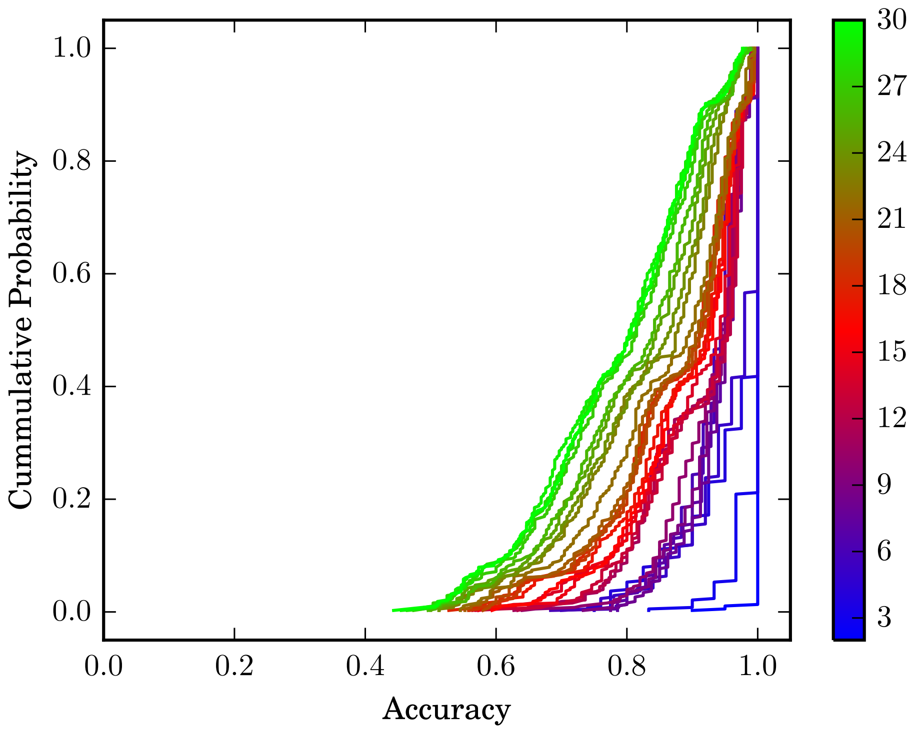

Figure 20. ECDF of classification accuracy by the size of suspects space starting from 2 authors, up to 30 authors, against Emirati tweets.

It can be seen in Figure 20 that, the larger the size of the suspects space, the more there are text vectorization methods that result in lower RFs AA classification accuracies. However, interestingly,

[12] http://stanfordnlp.github.io/CoreNLP/#human-languages-supported

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1513 even with the largest suspects space (which is 30 authors), there are at least certain configurations for
1514 the text vectorization method (which is the at least l-frequent k-skip n-grams) that enables the RFs AA
1515 models to achieve classification accuracies that are highly close to 1. More details on such successful
1516 configurations will be outlined in the next subsection of this evaluation.
1517 Figure 21 presents the classification accuracy as a function of the size of suspects space, across
1518 varying authors. This accuracy is measured by considering the performance of all of the feature
1519 extraction functions. It can be seen that the performance of solving AA problems against Emirati
1520 tweets is superior to those of Dutch and Greek datasets, and inferior to those of Spanish and US
1521 English.
1522 However, this is not necessarily an indication that solving AA problems is more difficult under
1523 Emirati tweets than Spanish or US English. This is due to the fact that some poorly performing features
1524 could degrade the overall classification accuracy, and mask the effect of the well performing features.

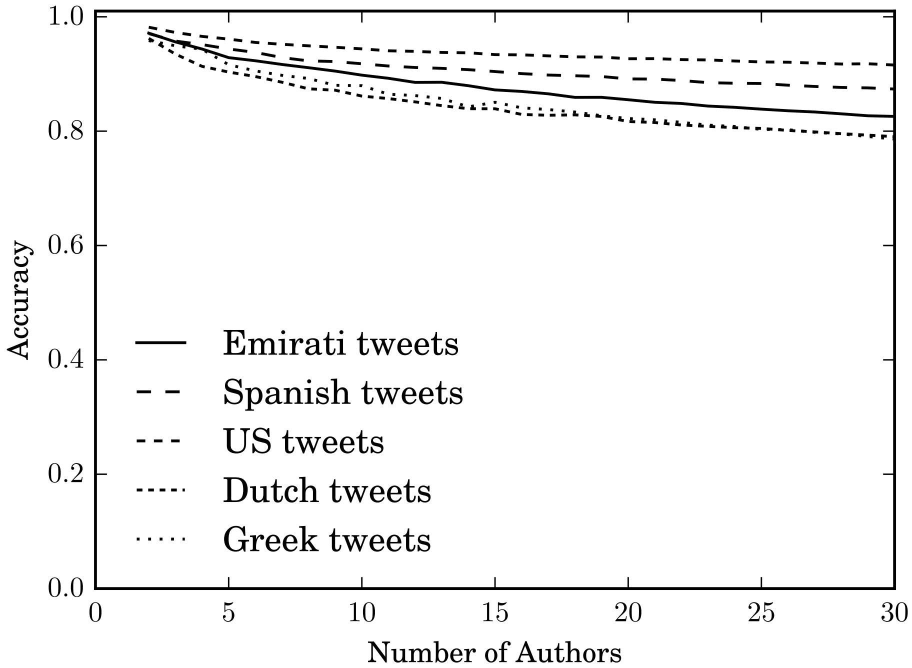

Figure 21. Classification accuracy as a function of the size of suspects space across varying languages.

1525 To demonstrate this, Figures 22, 23, 24 and 25 present the same results as those in Figure 21, except
1526 for choosing specific feature extraction functions that tend to perform well under specific datasets.

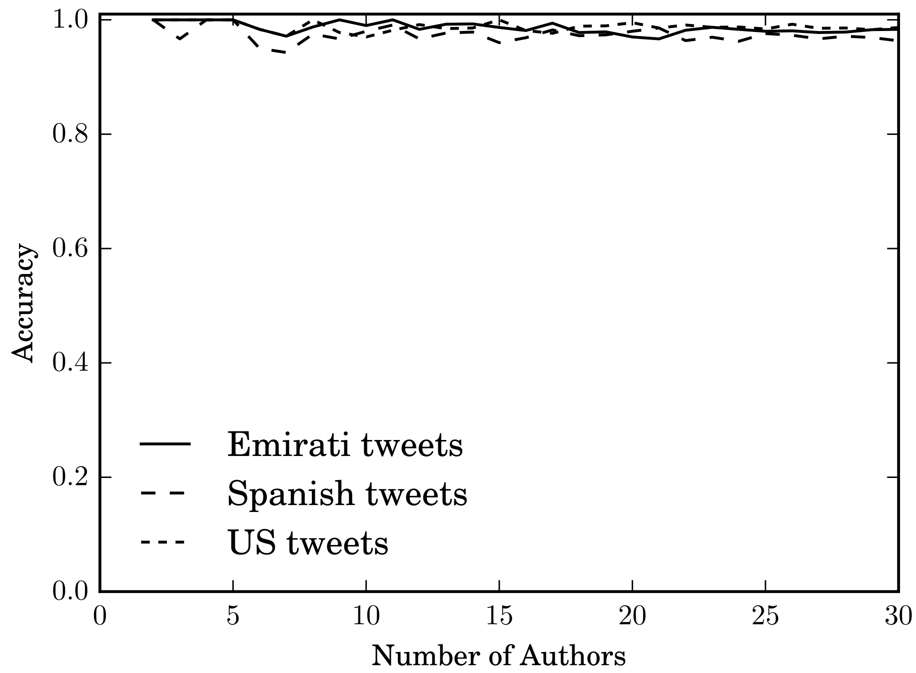

Figure 22. Classification accuracy as a function of the size of suspects space across varying languages, while using Gram = word-POS tag tuple, $l = 6$, $k = 0$, $n = 1$.

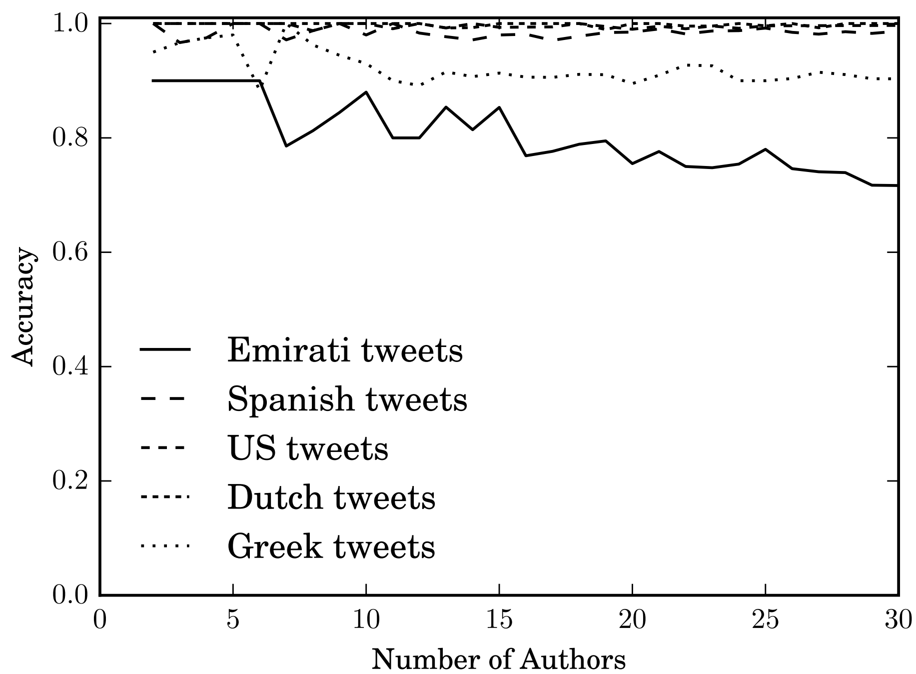

Figure 23. Classification accuracy as a function of the size of suspects space across varying languages, while using Gram = word, $l = 1$, $k = 0$, $n = 1$.

 doi:10.20944/preprints201708.0003.v1

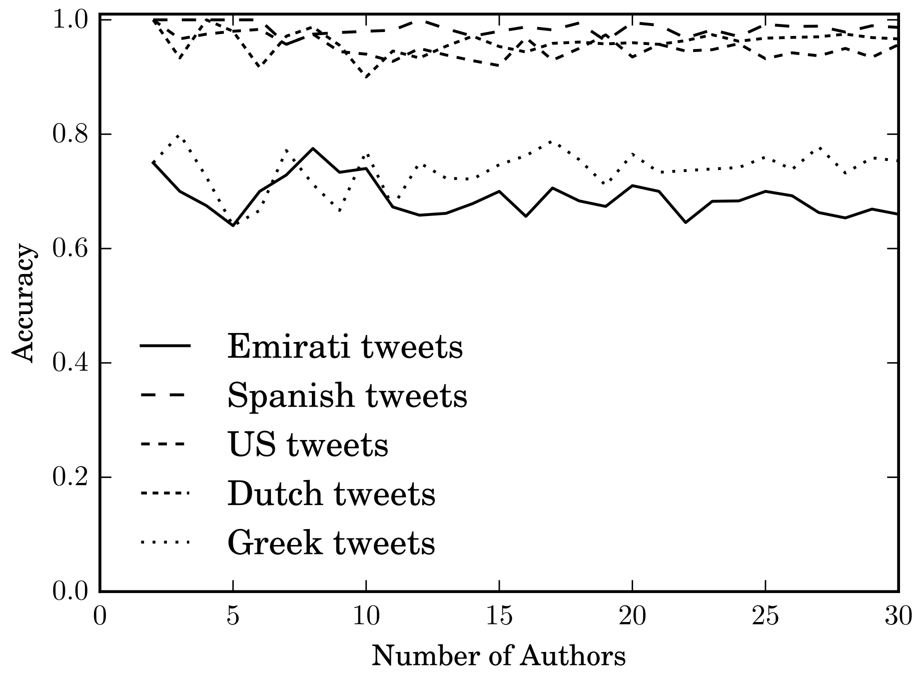

Figure 24. Classification accuracy as a function of the size of suspects space across varying languages, while using Gram = word, $l = 1, k = 0, n = 2$.

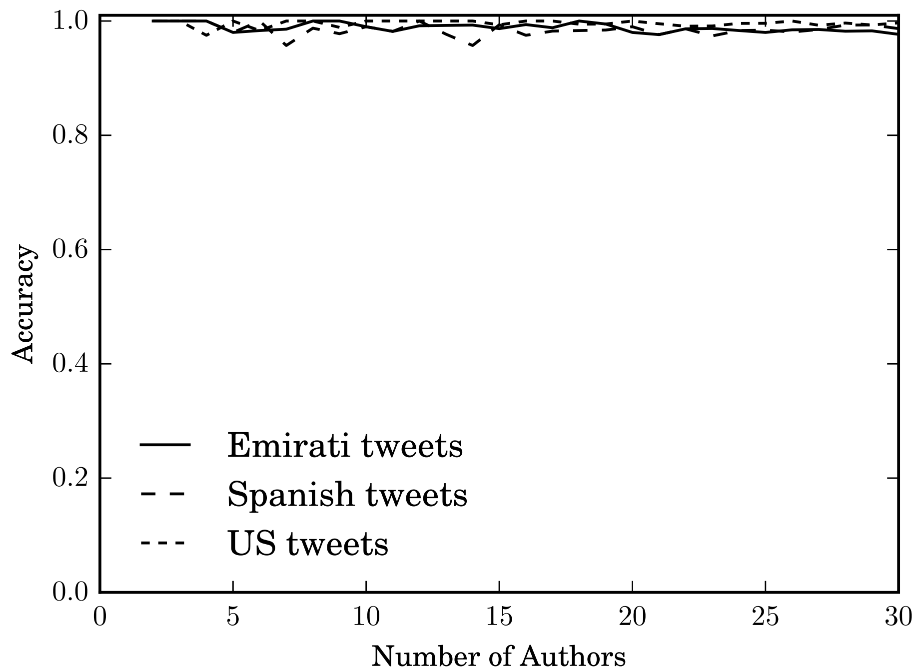

Figure 25. Classification accuracy as a function of the size of suspects space across varying languages, while using Gram = word-POS tag tuple, $l = 1, k = 0, n = 1$.

1527 It can be seen that the performance of solving AA problems with Emirati tweets can be highly
1528 similar to those of Spanish and US tweets datasets when certain feature extraction methods are chosen.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Namely, when defining grams as the tuple of word-POS tags. However, the performance on Emirati tweets degrades significantly when grams are defined to be words, as shown in Figures 23 and 24.

This suggests that, while the current methods of stylometry analysis were never previously evaluated against Emirati social media texts (and rarely against Arabic texts in general), accurately solving AA problems with Emirati tweets is nonetheless possible by using compound grams that are formed by combining successful feature extraction methods as found based on the literature of stylometry with respect to other languages.

11.2. Accuracy of Author Attribution Models as a Function of Text Vectorization Methods

Since this section discusses the effect of the various parameters of the feature extraction functions in a greater detail, the discussion is focused on Emirati tweets and a suspects space of 30 authors for brevity.

Figure 26 depicts the ECDFs of the evaluated RF AA classification models with varying values of l, when tested against a set of 30 suspect authors. The ECDFs generally indicate that the most accurate classification models can be identified when $l \in \{3, 6\}$. Interestingly, this is close to the value $l = 5$ that was found by Khonji et al. [64] for the other languages (i.e. Dutch, English, Greek, and Spanish).

However, the most accurate AA classification models under each value of $l \in \{1, 2, \ldots, 9\}$, are not significantly different than those found with different values of l. Table 19 presents the pair-wise statistical significance results against the most accurate models that are found under each value of l.

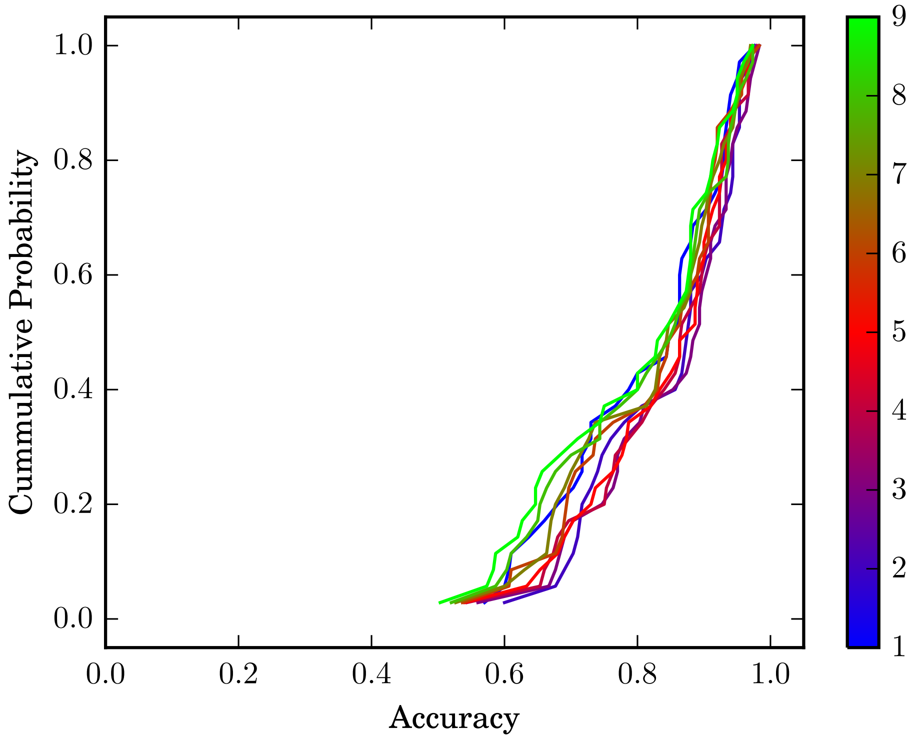

Figure 26. ECDF of classification accuracy for all $l \in \{1, 2, \ldots, 9\}$ with a suspects space of 30 authors using the Emirati tweets dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 19. The classification accuracy of correctly attributing 30 authors in the Emirati tweets dataset by using the most accurate model as categorized based on the value of l.

l	2 acc = 0.977	3 acc = 0.983	4 acc = 0.980	5 acc = 0.973	6 acc = 0.983	7 acc = 0.970	8 acc = 0.973	9 acc = 0.973
1 acc = 0.98	$p = 1.0000$ =	$p = 0.4346$ =	$p = 0.7932$ =	$p = 0.8002$ =	$p = 0.4585$ =	$p = 0.4595$ =	$p = 0.8042$ =	$p = 0.7752$ =
2 acc = 0.98	–	$p = 0.4456$ =	$p = 0.7682$ =	$p = 0.7972$ =	$p = 0.4665$ =	$p = 0.4306$ =	$p = 0.7872$ =	$p = 0.8202$ =
3 acc = 0.98	–	–	$p = 0.7672$ =	$p = 0.2717$ =	$p = 1.0000$ =	$p = 0.2048$ =	$p = 0.2837$ =	$p = 0.3147$ =
4 acc = 0.98	–	–	–	$p = 0.4665$ =	$p = 0.7952$ =	$p = 0.3257$ =	$p = 0.4665$ =	$p = 0.4885$ =
5 acc = 0.97	–	–	–	–	$p = 0.2897$ =	$p = 0.6743$ =	$p = 1.0000$ =	$p = 1.0000$ =
6 acc = 0.98	–	–	–	–	–	$p = 0.2088$ =	$p = 0.2717$ =	$p = 0.2887$ =
7 acc = 0.97	–	–	–	–	–	–	$p = 0.6643$ =	$p = 0.6603$ =
8 acc = 0.97	–	–	–	–	–	–	–	$p = 1.0000$ =

1547 Figure 27 depicts the ECDFs of the evaluated RF AA classification models with varying values of
1548 k. The ECDFs indicate that when $k = 0$ more accurate classification models can be identified than when
1549 $k = 1$, which suggests that the tolerating violations of the grams adjacency assumption is unhealthy for
1550 the purpose of identifying authors of Emirati social media texts. However, Table 20 indicates that the
1551 difference between the best performing classifiers under each value of k is not statistically significant.

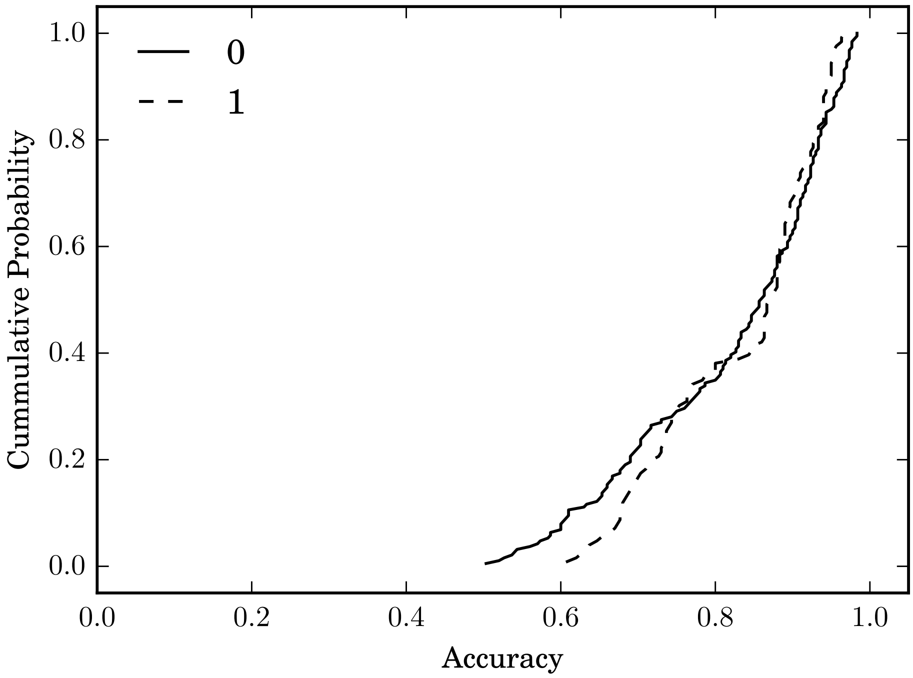

Figure 27. ECDF of classification accuracy by k with a suspects space of 30 authors using the Emirati tweets dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 20. The classification accuracy of correctly attributing 30 authors in the Emirati tweets dataset by using the most accurate model as categorized based on the value of k.

k	1 acc = 0.967
0 acc = 0.98	$p = 0.1279$ =

Figure 28 depicts the ECDFs of the evaluated RF AA classification models with varying values of n. The ECDFs indicate that when $n = 1$ more accurate classification models can be identified than when $n > 1$, which suggests that observing the distribution of grams in relation to their adjacent ones is unhealthy for the purpose of identifying authors of Emirati social media texts. However, Table 21 indicates that the difference in accuracy between the most accurate models under each value of n is not statistically significant. The only exception is between the cases when $n = 1$ and $n = 3$ by which the difference in accuracy is statistically significant.

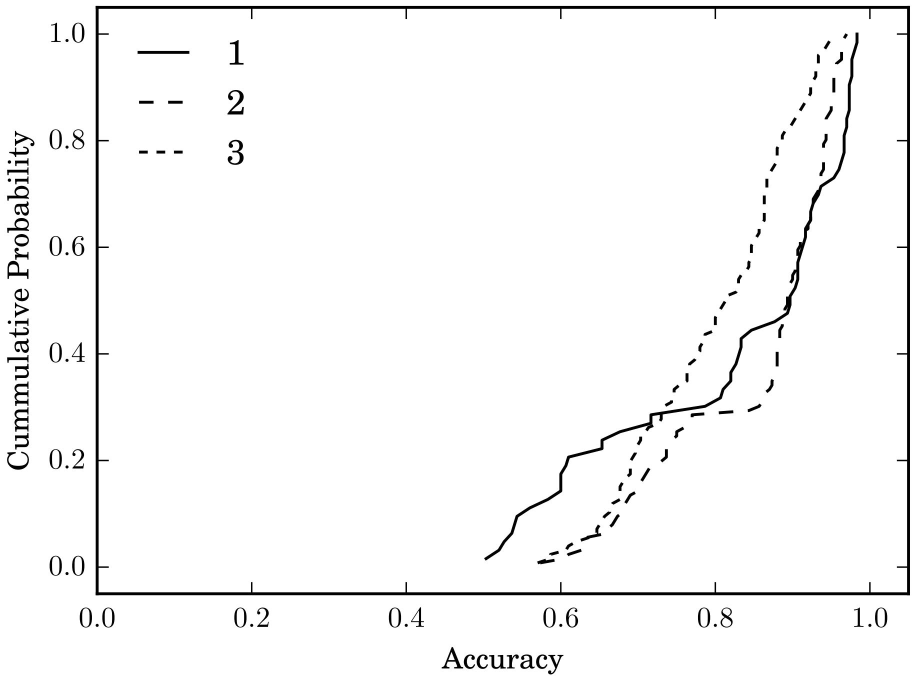

Figure 28. ECDF of classification accuracy by n with a suspects space of 30 authors using the Emirati tweets dataset.

Table 21. The classification accuracy of correctly attributing 30 authors in the Emirati tweets dataset by using the most accurate model as categorized based on the value of n.

n	2 acc = 0.970	3 acc = 0.953
1 acc = 0.98	$p = 0.2168$ =	$p = 0.0340$ *
2 acc = 0.97	–	$p = 0.1828$ =

Figure 29 depicts the ECDFs of the evaluated RF AA classification models with varying definitions of grams. The ECDFs indicate that, compound grams allow for the identification of more accurate

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1561 classification models than otherwise. Table 22 indicates that the increase in classification accuracy
1562 with the most accurate models under compound grams is always statistically significant. The only
1563 exception is the compound gram pos-dep which failed to demonstrate statistical significance against
1564 the gram pos ($p = 0.0789$), and failed to be more accurate than the gram pos.

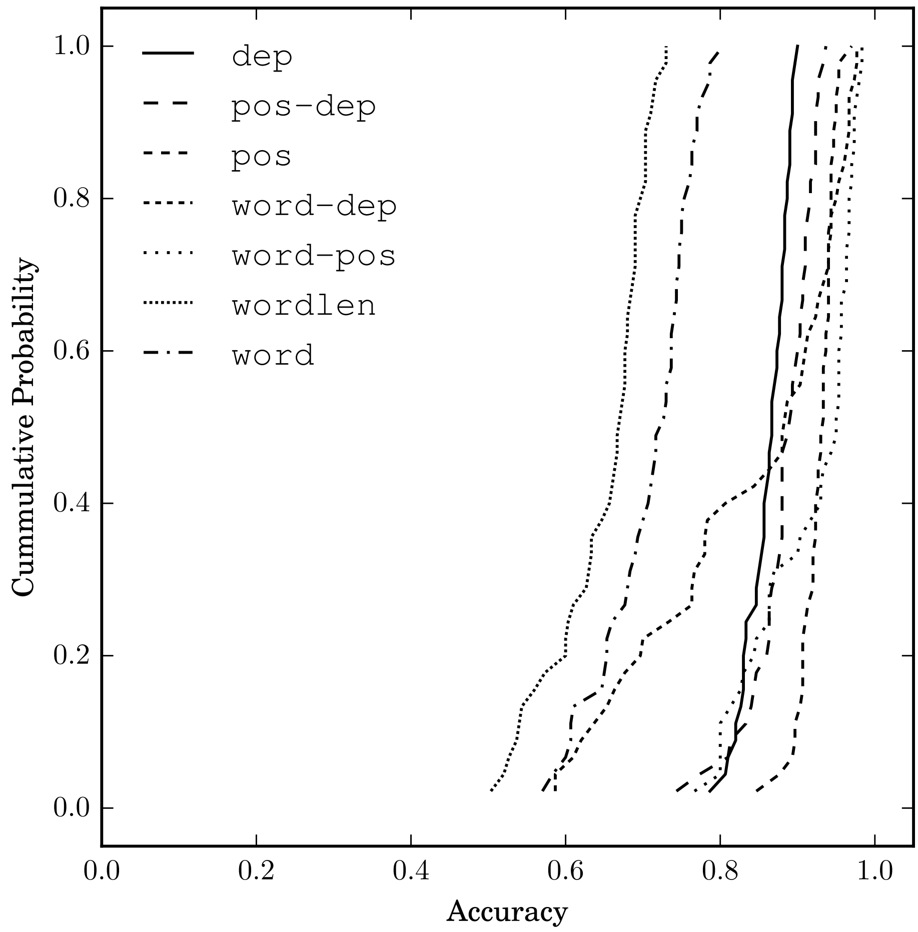

Figure 29. ECDF of classification accuracy by gram with a suspects space of 30 authors using the Emirati tweets dataset.

 doi:10.20944/preprints201708.0003.v1

Table 22. The classification accuracy of correctly attributing 30 authors in the Emirati tweets dataset by using the most accurate model as categorized based on the definition of grams.

Gram	`pos-dep` acc $= 0.937$	`pos` acc $= 0.970$	`word-dep` acc $= 0.977$	`word-pos` acc $= 0.983$	`wordlen` acc $= 0.730$	`word` acc $= 0.803$
`dep` acc $= 0.90$	$p = 0.0789$ =	$p = 0.0030$ **	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***
`pos-dep` acc $= 0.94$	–	$p = 0.0340$ *	$p = 0.0100$ **	$p = 0.0030$ **	$p = 0.0010$ ***	$p = 0.0010$ ***
`pos` acc $= 0.97$	–	–	$p = 0.4316$ =	$p = 0.2348$ =	$p = 0.0010$ ***	$p = 0.0010$ ***
`word-dep` acc $= 0.98$	–	–	–	$p = 0.4545$ =	$p = 0.0010$ ***	$p = 0.0010$ ***
`word-pos` acc $= 0.98$	–	–	–	–	$p = 0.0010$ ***	$p = 0.0010$ ***
`wordlen` acc $= 0.73$	–	–	–	–	–	$p = 0.0280$ *

Table 23 presents a classification accuracy ranked list and parameters of the text vectorization methods of the best performing classifiers that the difference between classification accuracies are small enough to not be statistically significant. It can be seen that the top 10 best performing classifiers exclusively make use of compound grams. This suggests that our novel definition of grams is successful in allowing for the achievement of higher classification accuracies under the Emirati tweets domain than when grams are defined classically.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Table 23. The top 45 most accurate classifiers when the space of suspects is 30 authors using the Emirati tweets dataset.

l	k	n	Gram	Accuracy
6	0	1	word-pos	0.9833
3	0	1	word-pos	0.9833
4	0	1	word-pos	0.9800
3	0	1	word-dep	0.9767
2	0	1	word-dep	0.9767
1	0	1	word-pos	0.9767
9	0	1	word-pos	0.9733
8	0	1	word-pos	0.9733
8	0	1	word-dep	0.9733
5	0	1	word-pos	0.9733
3	0	2	pos	0.9700
7	0	1	word-pos	0.9700
2	0	1	word-pos	0.9700
4	1	2	word-pos	0.9667
4	0	2	word-pos	0.9667
3	0	2	word-pos	0.9667
7	0	1	word-dep	0.9667
5	0	1	word-dep	0.9667
4	0	1	word-dep	0.9667
8	0	2	word-pos	0.9633
5	1	2	word-pos	0.9633
3	1	2	word-pos	0.9633
6	0	1	word-dep	0.9633
9	0	1	word-dep	0.9600
6	1	2	word-pos	0.9567
6	0	2	word-pos	0.9567
2	0	2	word-pos	0.9567
7	1	2	word-pos	0.9533
7	0	2	word-pos	0.9533
5	0	2	word-pos	0.9533
2	1	3	pos	0.9533
2	1	2	word-pos	0.9533
2	0	2	pos	0.9533
1	0	1	word-dep	0.9533
9	1	2	word-pos	0.9500
9	1	2	pos	0.9500
8	1	2	word-pos	0.9500
3	1	2	word-dep	0.9500
1	0	3	pos	0.9500
8	1	3	pos	0.9467
7	1	2	pos	0.9467
9	0	2	word-pos	0.9433
7	0	2	pos	0.9433
4	0	2	pos	0.9433
3	1	2	pos	0.9433

However, it is important to note that an accurately performing AA classifier is not necessarily an indication of model's ability in identifying the writing styles of authors. For example, if the dataset contains a significant author-topic bias, then a model that is originally intended to be an author identification model, can possibly be partly both, an author identification model, as well as a topic identification model. Therefore care must be taken to ensure that the used features do not contain too much topic information, as such topic information could confuse the learning algorithm and transform it into a topic classifier up to a larger degree than otherwise. This is specifically a concern when features that contain words are used, as such words could be content words (as opposed to function words).

Therefore, to ensure that the reason our best performing compound gram (*word*-*POS* or `word-pos` as shown in Table 23) is not so due to it containing excessive topic information that leads the model to be partly a topic classifier, the listing below presents the list of the 20 most important features as aggregated from each of the 10 evaluation folds as used in our RF models (duplicate entries are removed):

- "-PUNC.
- #-NN.
- #-VBD.
- ,-PUNC.
- .-PUNC.
- ..-PUNC.
- ...-PUNC.
- :-PUNC.
- ?-NN.
- @-NN.
- @-VBD.
- NAME-NN.
- TAG-NN.
- http://URL-NN.
- الله-NNP.
- ان-IN.
- ب-IN.
- في-IN.
- ل-IN.
- لا-RP.
- ما-WP.
- من-IN.
- ه-PRP.
- ه-PRP$.
- و-CC.
- يا-RP.

It can be seen that all of the listed compound grams are free of content or topic words. Interestingly, the identified Arabic words in the list above are also Arabic function words. The only arguable word is "الله", which translates to "*God*". However, since the word "الله" is often used in various expressions that are independent of the topic, we believe that it is fair to consider it a word that does not contain significant topic information.

On the other hand, the least performing features (i.e. features that contribute least to AA model's decision in solving AA problems) contain a significant amount of content or topic words. A list of such features is presented below:

- 1,990,0-CD.
- 587,0-CD.
- 6,0-CD.
- 800-CD.
- are-VBD.
- mistake-NN.
- realize-NN.
- truth-NN.
- اتمنا-VBD.
- اظن-VBP.
- اقدر-VBP.
- التكفير-DTNN.
- الجبال-DTNN.
- الجمال-DTNN.
- الراي-DTNN.
- القطاعات-DTNNS.
- الكثير-DTNN.
- الهوى-DTNN.
- الوكيل-DTNN.
- تاون-NNP.
- تبد-VBP.
- توقيت-NN.
- حسابي-JJ.
- دونيس-NNP.
- راقية-JJ.
- زيورخ-NNP.
- ضفاف-NN.
- طعام-NN.
- فاعف-JJ.
- متعدد-JJ.
- مريض-NN.
- معيشة ،-NN.
- يضيف-VBP.

This supports the claims that the KIT-30 dataset does not contain significant author-topic bias, and that the proposed compound grams are likely helping the learning algorithm to actually find author identification models, as opposed to topic classification models.

12. Discussion

This paper introduced electronic text stylometry problems under a unified notation in probability terms, their importance in enhancing various upper layer applications, and the key challenges that are currently faced in this field. Such challenges include the optimization of the stylometry problem solvers to maximize their classification accuracy, performing accurate stylometry analysis across distinct domains, the generalization of existing data representation methods to enhance our understanding as well as potentially identify novel ones, the construction of novel evaluation datasets for certain domains, such as Emirati social media texts, as well as the availability of software to facilitate convenient reproducibility.

This paper has also moved towards addressing some of the key challenges that the current state of the literature on stylometry faces, such as 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 23 thousand feature extraction functions, which evaluated them under the same

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

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-grams-based functions proved to be superior among the more sophisticated variants (i.e. syntactic n-grams with dependency trees, and k-skipped n-grams).

Another key contribution of this paper relates to the uncertainty that is associated with the applicability of the stylometry problem solvers against the domain of Emirati Arabic texts. To work towards addressing this issue, we have constructed the KIT-30 dataset, which is the first Emirati social media author identification evaluation dataset. Interestingly, our studies found that the scalability issues of AA problem solvers that are generally reported in the literature with respect to the size of the suspect authors space, is noticeably more aggressive than what our findings indicate. For example, we were able to achieve a classification accuracy of over 0.98 when solving AA problems that were constructed based on chunks of Emirati tweets, with a set of 30 suspect authors. This accuracy is notably higher than evaluated in the literature on similar space sizes of suspect authors in literature [28,68], specially when knowing that our chunks of tweets, per author, remained relatively small (only a few hundreds of tweets per chunk).

Additionally, in order to work towards addressing the lack of conveniently-available implementations of stylometry methods, we have developed an extensive electronic text feature extraction library, with a highly intuitive API. This library offers, by far, the most extensive set of electronic text stylometry feature extraction methods to date, which is partly thanks to our generalization of n-gram-based feature extraction methods. The library also contains a number of novelties, such as novel definitions of grams (e.g. compound grams) for both, n-grams-based methods, as well as CFG rewrite-rules. Interestingly, when using our feature extraction library, our evaluation of state of the art AA solver against Emirati tweets AA problems indicate that the use of compound grams allow for the identification of more accurate AA models, than the alternative case when conventional definitions of grams are followed.

Thanks to our generalization of n-gram-based feature extraction functions, which is also implemented in our feature extraction library (Fextractor), the problem of identifying better *direction* can be represented as the problem of identifying optimum grams matrix (as discussed in Section 5). This is more generic than the use of dependency or constituency trees (as used by syntactic n-grams), since such gram matrices could be initialized in such ways that cycles, or repetitions, are permitted. In other words, once such constraints are removed (i.e. cycles and repetitions are allowed), could we identify a grams matrix initialization algorithm, that ultimately leads into a higher AV classification accuracy?

Supplementary Materials: The feature extraction library, Fextractor, is available at https://gitlab.com/mmaakh/fextractor.git. The evaluation code is available at https://gitlab.com/mmaakh/stylometry-survey-evaluation.git. The KIT-30 dataset is available at https://gitlab.com/mmaakh/kit-30.git.

Acknowledgments: Thanks to Buhooth for funding this work. The authors of this paper are also highly grateful to the Khalifa University students, Zainab Al-Blooshi and Fatima Al-Hammadi, for their generous help in identifying Emirati Twitter accounts for the purpose of extending the KIT-30 dataset. Special thanks to Dr. Efstathios Stamatatos for his generous guidance and the fruitful discussions that significantly enhanced our understanding about stylometry problems, as well as our understanding about the desirable properties that should be satisfied in their evaluation problem sets. The help we received from Dr. Efstathios Stamatatos have greatly helped us enhance our datasets, problem sets and evaluation methodologies.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AA	Author Attribution

 doi:10.20944/preprints201708.0003.v1

AC	Author Clustering
AD	Author Diarization
ANN	Artificial Neural Network
AP	Author Profiling
AR	Approximate Randomization
AV	Author Verification
CFG	Context-Free Grammar
CNG	Common n-grams
DA	Domain Adaptation
ECDF	empirical commutative density function
IG	Information Gain
KDE	Kernel Density Estimation
MCC	Multi-domain Closed-set Classification
MOC	Multi-domain Open-set Classification
PDF	probability density function
POS	Part of Speech
RF	Random Forest
RNN	Recurrent Neural Network
SCC	Single-domain Closed-set Classification
SOC	Single-domain Open-set Classification
SVM	Support Vector Machine
UAE	United Arab Emirates

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Appendix A Data Representation Evaluation Tables

Table A1. p values of the most accurate methods from the perspective of grams (S24 problems set).

	funcword acc = 0.88	pos-dep acc = 0.83	pos acc = 0.92	word-dep acc = 0.79	word-pos acc = 0.79	wordlen acc = 0.67	wordshape-word acc = 0.79	wordshape acc = 0.71	word acc = 0.88
dep acc = 0.83	$p = 0.7313$ =	$p = 1.0000$ =	$p = 0.2957$ =	$p = 0.4835$ =	$p = 0.5405$ =	$p = 0.1129$ =	$p = 0.5315$ =	$p = 0.2228$ =	$p = 0.6983$ =
funcword acc = 0.88	–	$p = 0.7173$ =	$p = 0.6863$ =	$p = 0.3497$ =	$p = 0.3407$ =	$p = 0.0979$ =	$p = 0.3516$ =	$p = 0.1638$ =	$p = 1.0000$ =
pos-dep acc = 0.83	–	–	$p = 0.2318$ =	$p = 0.4915$ =	$p = 0.5265$ =	$p = 0.1039$ =	$p = 0.5764$ =	$p = 0.2138$ =	$p = 0.7303$ =
pos acc = 0.92	–	–	–	$p = 0.1958$ =	$p = 0.1768$ =	$p = 0.0360$ *	$p = 0.2428$ =	$p = 0.0979$ =	$p = 0.7003$ =
word-dep acc = 0.79	–	–	–	–	$p = 1.0000$ =	$p = 0.1668$ =	$p = 1.0000$ =	$p = 0.3656$ =	$p = 0.3357$ =
word-pos acc = 0.79	–	–	–	–	–	$p = 0.1718$ =	$p = 1.0000$ =	$p = 0.3836$ =	$p = 0.3357$ =
wordlen acc = 0.67	–	–	–	–	–	–	$p = 0.2368$ =	$p = 0.5365$ =	$p = 0.0699$ =
wordshape-word acc = 0.79	–	–	–	–	–	–	–	$p = 0.3576$ =	$p = 0.3477$ =
wordshape acc = 0.71	–	–	–	–	–	–	–	–	$p = 0.1359$ =

Table A2. p values of the most accurate methods from the perspective of grams (S1000 problems set).

	funcword acc = 0.69	pos-dep acc = 0.81	pos acc = 0.83	word-dep acc = 0.85	word-pos acc = 0.85	wordlen acc = 0.54	wordshape-word acc = 0.84	wordshape acc = 0.71	word acc = 0.79
dep acc = 0.79	$p = 0.0010$ ***	$p = 0.0559$ =	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 1.0000$ =
funcword acc = 0.69	–	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.2527$ =	$p = 0.0010$ ***
pos-dep acc = 0.81	–	–	$p = 0.0090$ **	$p = 0.0020$ **	$p = 0.0020$ **	$p = 0.0010$ ***	$p = 0.0120$ *	$p = 0.0010$ ***	$p = 0.0759$ =
pos acc = 0.83	–	–	–	$p = 0.1449$ =	$p = 0.0619$ =	$p = 0.0010$ ***	$p = 0.2318$ =	$p = 0.0010$ ***	$p = 0.0010$ ***
word-dep acc = 0.85	–	–	–	–	$p = 0.4236$ =	$p = 0.0010$ ***	$p = 0.3766$ =	$p = 0.0010$ ***	$p = 0.0010$ ***
word-pos acc = 0.85	–	–	–	–	–	$p = 0.0010$ ***	$p = 0.1618$ =	$p = 0.0010$ ***	$p = 0.0010$ ***
wordlen acc = 0.54	–	–	–	–	–	–	$p = 0.0010$ ***	$p = 0.0010$ ***	$p = 0.0010$ ***
wordshape-word acc = 0.84	–	–	–	–	–	–	–	$p = 0.0010$ ***	$p = 0.0010$ ***
wordshape acc = 0.71	–	–	–	–	–	–	–	–	$p = 0.0010$ ***

Table A3. Most accurate feature extraction methods (S24 problems set).

Direction	l	k	n	Gram	Accuracy
spatial	2	0	2	pos	0.9167
spatial	1	1	2	pos	0.9167
spatial	1	0	2	pos	0.9167
spatial	7	1	2	pos	0.8750
spatial	5	0	2	pos	0.8750
spatial	4	0	2	pos	0.8750
spatial	3	1	2	pos	0.8750
spatial	2	2	2	pos	0.8750
spatial	1	3	2	pos	0.8750
spatial	1	2	2	pos	0.8750
N/A	2	0	1	word	0.8750

N/A	1	0	1	funcword	0.8750
spatial	6	0	2	pos	0.8333
spatial	5	2	2	dep	0.8333
spatial	4	2	2	pos	0.8333
spatial	3	3	2	dep	0.8333
spatial	3	2	2	dep	0.8333
spatial	3	0	2	pos	0.8333
spatial	2	3	3	dep	0.8333
spatial	2	2	2	dep	0.8333
spatial	2	1	2	pos	0.8333
spatial	2	0	3	dep	0.8333
spatial	1	3	2	dep	0.8333
spatial	1	0	2	dep	0.8333
N/A	12	0	1	funcword	0.8333
N/A	7	0	1	word	0.8333
N/A	6	0	1	funcword	0.8333
N/A	5	0	1	pos-dep	0.8333
N/A	3	0	1	word	0.8333
N/A	2	0	1	funcword	0.8333
spatial	44	0	2	pos-dep	0.7917
spatial	18	0	2	pos	0.7917
spatial	15	0	2	pos-dep	0.7917
spatial	14	0	2	pos	0.7917
spatial	14	0	2	pos-dep	0.7917
spatial	13	0	2	dep	0.7917
spatial	12	2	2	pos-dep	0.7917
spatial	12	0	2	pos	0.7917
spatial	10	3	2	pos-dep	0.7917
spatial	9	3	2	pos	0.7917
spatial	9	3	2	pos-dep	0.7917
spatial	9	2	2	pos	0.7917
spatial	9	2	2	pos-dep	0.7917
spatial	8	2	2	pos	0.7917
spatial	8	0	3	dep	0.7917

Table A4. Most accurate feature extraction methods (S1000 problems set).

Direction	l	k	n	Gram	Accuracy
N/A	2	0	1	word-pos	0.8510
N/A	2	0	1	word-dep	0.8460
N/A	3	0	1	wordshape-word	0.8410
spatial	2	3	2	pos	0.8320
spatial	2	2	2	pos	0.8310
N/A	3	0	1	word-pos	0.8250
spatial	2	1	2	pos	0.8160
N/A	3	0	1	word-dep	0.8140
N/A	4	0	1	wordshape-word	0.8120
N/A	2	0	1	pos-dep	0.8100
spatial	2	0	2	pos	0.8060

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

spatial	3	3	2	pos	0.8030
spatial	2	3	2	pos-dep	0.8030
N/A	4	0	1	word-pos	0.8000
spatial	2	3	3	pos	0.7990
spatial	3	2	2	pos	0.7910
N/A	5	0	1	word-pos	0.7910
spatial	2	0	2	dep	0.7900
N/A	6	0	1	word	0.7900
spatial	4	3	2	pos	0.7870
spatial	2	2	2	pos-dep	0.7870
spatial	3	1	2	pos	0.7860
N/A	6	0	1	word-pos	0.7850
N/A	3	0	1	pos-dep	0.7830
spatial	3	3	2	dep	0.7790
spatial	2	3	2	dep	0.7790
spatial	2	2	3	pos	0.7790
spatial	2	2	2	dep	0.7790
spatial	2	1	2	pos-dep	0.7780
N/A	4	0	1	pos	0.7780
spatial	2	1	2	dep	0.7770
N/A	7	0	1	wordshape-word	0.7770
N/A	4	0	1	pos-dep	0.7760
N/A	2	0	1	pos	0.7760
spatial	2	3	2	word-pos	0.7750
N/A	5	0	1	pos-dep	0.7750
N/A	4	0	1	word-dep	0.7710
spatial	2	0	2	pos-dep	0.7700
N/A	5	0	1	pos	0.7700
spatial	4	1	2	pos	0.7690
spatial	3	3	2	pos-dep	0.7690
N/A	2	0	1	word	0.7690
spatial	4	2	2	pos	0.7680
spatial	3	2	2	dep	0.7660
spatial	3	1	2	dep	0.7640

Table A5. Least accurate feature extraction methods (S24 problems set).

Direction	l	k	n	Gram	Accuracy
deptree	32	0	3	word	0.0833
deptree	32	0	3	wordshape-word	0.0833
deptree	32	0	3	word-pos	0.0833
deptree	31	3	3	pos-dep	0.0833
deptree	31	2	3	pos-dep	0.0833
deptree	31	0	3	word	0.0833
deptree	31	0	3	wordshape-word	0.0833
deptree	31	0	3	word-pos	0.0833
deptree	30	2	3	pos-dep	0.0833
deptree	30	0	3	word	0.0833
deptree	30	0	3	wordshape-word	0.0833

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

deptree	30	0	3	word-pos	0.0833
deptree	29	1	3	word	0.0833
deptree	29	1	3	wordshape-word	0.0833
deptree	29	1	3	pos-dep	0.0833
deptree	28	1	3	word	0.0833
deptree	28	1	3	wordshape-word	0.0833
deptree	27	1	3	word	0.0833
deptree	27	1	3	wordshape-word	0.0833
deptree	26	1	3	word	0.0833
deptree	26	1	3	wordshape-word	0.0833
deptree	25	1	3	word	0.0833
deptree	21	3	2	word-pos	0.0833
deptree	11	0	3	pos-dep	0.0833
deptree	1	1	3	word-pos	0.0833
deptree	1	1	3	word-dep	0.0833
deptree	1	0	3	wordshape-word	0.0833
spatial	10	1	3	word	0.0417
spatial	10	1	3	wordshape-word	0.0417
spatial	9	1	3	word	0.0417
spatial	9	1	3	wordshape-word	0.0417
spatial	3	0	3	word	0.0417
spatial	3	0	3	wordshape-word	0.0417
deptree	99	3	3	pos-dep	0.0417
deptree	99	1	3	wordshape	0.0417
deptree	98	1	3	wordshape	0.0417
deptree	85	3	3	pos-dep	0.0417
deptree	62	0	3	wordlen	0.0417
deptree	60	0	3	wordlen	0.0417
deptree	59	0	3	wordlen	0.0417
deptree	58	0	3	wordlen	0.0417
deptree	57	0	3	wordlen	0.0417
deptree	42	1	3	pos-dep	0.0417
deptree	31	1	3	pos-dep	0.0417
deptree	30	1	3	pos-dep	0.0417

Table A6. Least accurate feature extraction methods (S1000 problems set).

Direction	l	k	n	Gram	Accuracy
deptree	34	2	2	word-pos	0.1240
deptree	34	2	2	word-dep	0.1240
deptree	33	3	2	word-pos	0.1240
deptree	33	3	2	word-dep	0.1240
deptree	33	2	2	word-pos	0.1240
deptree	33	2	2	word-dep	0.1240
deptree	6	0	3	wordshape-word	0.1240
deptree	3	0	3	word-pos	0.1240
N/A	71	0	1	funcword	0.1240
N/A	63	0	1	word-pos	0.1240
N/A	63	0	1	word-dep	0.1240

N/A	62	0	1	word-pos	0.1240
N/A	62	0	1	word-dep	0.1240
N/A	61	0	1	word-pos	0.1240
N/A	61	0	1	word-dep	0.1240
N/A	60	0	1	word-pos	0.1240
N/A	60	0	1	word-dep	0.1240
N/A	55	0	1	word	0.1240
N/A	55	0	1	wordshape-word	0.1240
N/A	55	0	1	funcword	0.1240
N/A	54	0	1	word	0.1240
N/A	54	0	1	wordshape-word	0.1240
spatial	98	3	3	pos	0.1230
spatial	97	3	3	pos	0.1230
spatial	56	2	3	pos	0.1230
spatial	55	2	3	pos	0.1230
spatial	54	2	3	pos	0.1230
spatial	53	2	3	pos	0.1230
spatial	48	2	2	pos-dep	0.1230
spatial	47	2	2	pos-dep	0.1230
spatial	40	0	2	pos-dep	0.1230
spatial	39	0	2	pos-dep	0.1230
spatial	33	3	2	funcword	0.1230
spatial	33	0	3	dep	0.1230
spatial	32	0	3	dep	0.1230
spatial	31	0	3	dep	0.1230
spatial	25	0	3	pos-dep	0.1230
spatial	16	3	3	word-pos	0.1230
spatial	16	3	3	word-dep	0.1230
spatial	15	3	3	word-pos	0.1230
spatial	15	3	3	word-dep	0.1230
deptree	6	0	3	word-pos	0.1230
spatial	29	0	3	pos	0.1220
spatial	28	0	3	pos	0.1220
spatial	24	0	3	pos-dep	0.1220

Table A7. Least accurate feature extraction methods while given the constraint that $l \leq 5$ (S24 problems set).

Direction	l	k	n	Gram	Accuracy
spatial	5	0	3	word	0.1250
spatial	4	0	3	word	0.1250
spatial	4	0	3	wordshape-word	0.1250
spatial	2	0	3	word	0.1250
spatial	2	0	3	wordshape-word	0.1250
deptree	5	3	3	word-pos	0.1250
deptree	5	3	3	funcword	0.1250
deptree	5	3	2	funcword	0.1250
deptree	5	2	3	wordshape-word	0.1250
deptree	5	2	3	word-dep	0.1250
deptree	5	2	3	funcword	0.1250

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1731

deptree	5	1	3	funcword	0.1250
deptree	5	0	3	funcword	0.1250
deptree	5	0	2	funcword	0.1250
deptree	4	3	3	word-dep	0.1250
deptree	4	3	3	funcword	0.1250
deptree	4	2	3	funcword	0.1250
deptree	4	2	2	funcword	0.1250
deptree	4	1	3	word-dep	0.1250
deptree	4	1	3	funcword	0.1250
deptree	4	0	3	funcword	0.1250
deptree	3	3	3	funcword	0.1250
deptree	3	2	3	funcword	0.1250
deptree	3	1	3	funcword	0.1250
deptree	3	0	3	funcword	0.1250
deptree	3	0	2	funcword	0.1250
deptree	2	3	3	funcword	0.1250
deptree	2	2	3	funcword	0.1250
deptree	2	1	3	funcword	0.1250
deptree	2	0	3	funcword	0.1250
deptree	1	3	3	word-pos	0.1250
deptree	1	3	3	word-dep	0.1250
deptree	1	3	3	funcword	0.1250
deptree	1	2	3	funcword	0.1250
deptree	1	1	3	funcword	0.1250
deptree	1	0	3	word	0.1250
deptree	1	0	3	word-pos	0.1250
deptree	1	0	3	word-dep	0.1250
deptree	1	0	3	funcword	0.1250
spatial	1	3	3	word	0.0833
deptree	1	1	3	word-pos	0.0833
deptree	1	1	3	word-dep	0.0833
deptree	1	0	3	wordshape-word	0.0833
spatial	3	0	3	word	0.0417
spatial	3	0	3	wordshape-word	0.0417

Table A8. Least accurate feature extraction methods while given the constraint that $l \leq 5$ (S1000
1732 problems set).

1733

Direction	l	k	n	Gram	Accuracy
deptree	5	0	3	word	0.1250
deptree	5	0	3	wordshape-word	0.1250
deptree	5	0	3	word-pos	0.1250
deptree	5	0	3	word-dep	0.1250
deptree	5	0	3	funcword	0.1250
deptree	4	3	3	word	0.1250
deptree	4	3	3	word-pos	0.1250
deptree	4	3	3	word-dep	0.1250
deptree	4	3	3	funcword	0.1250
deptree	4	2	3	word	0.1250
deptree	4	2	3	wordshape-word	0.1250
deptree	4	2	3	word-pos	0.1250

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1734

deptree	4	2	3	word-dep	0.1250
deptree	4	2	3	funcword	0.1250
deptree	4	1	3	word	0.1250
deptree	4	1	3	wordshape-word	0.1250
deptree	4	1	3	word-pos	0.1250
deptree	4	1	3	word-dep	0.1250
deptree	4	1	3	funcword	0.1250
deptree	4	0	3	word	0.1250
deptree	4	0	3	wordshape-word	0.1250
deptree	4	0	3	word-pos	0.1250
deptree	4	0	3	word-dep	0.1250
deptree	4	0	3	funcword	0.1250
deptree	3	3	3	word	0.1250
deptree	3	3	3	wordshape-word	0.1250
deptree	3	3	3	word-pos	0.1250
deptree	3	3	3	word-dep	0.1250
deptree	3	3	3	funcword	0.1250
deptree	3	2	3	word	0.1250
deptree	3	2	3	wordshape-word	0.1250
deptree	3	2	3	word-pos	0.1250
deptree	3	2	3	word-dep	0.1250
deptree	3	2	3	funcword	0.1250
deptree	3	1	3	word	0.1250
deptree	3	1	3	wordshape-word	0.1250
deptree	3	1	3	word-pos	0.1250
deptree	3	1	3	word-dep	0.1250
deptree	3	1	3	funcword	0.1250
deptree	3	0	3	word	0.1250
deptree	3	0	3	funcword	0.1250
deptree	2	3	3	word-dep	0.1250
deptree	2	2	3	word-dep	0.1250
deptree	2	1	3	word-dep	0.1250
deptree	3	0	3	word-pos	0.1240

Table A9. Cluster of feature extraction methods that their differences against the most accurate method
1735 are not statistically significant ($p > 0.05$) (S24 problems set).

1736

Direction	l	k	n	Gram	Accuracy
spatial	2	0	2	pos	0.9167
spatial	1	1	2	pos	0.9167
spatial	1	0	2	pos	0.9167
spatial	7	1	2	pos	0.8750
spatial	5	0	2	pos	0.8750
spatial	4	0	2	pos	0.8750
spatial	3	1	2	pos	0.8750
spatial	2	2	2	pos	0.8750
spatial	1	3	2	pos	0.8750
spatial	1	2	2	pos	0.8750
N/A	2	0	1	word	0.8750

N/A	1	0	1	word	0.8750
spatial	6	0	2	pos	0.8333
spatial	5	2	2	dep	0.8333
spatial	4	2	2	pos	0.8333
spatial	3	3	2	dep	0.8333
spatial	3	2	2	dep	0.8333
spatial	3	0	2	pos	0.8333
spatial	2	3	3	dep	0.8333
spatial	2	2	2	dep	0.8333
spatial	2	1	2	pos	0.8333
spatial	2	0	3	dep	0.8333
spatial	1	3	2	dep	0.8333
spatial	1	0	2	dep	0.8333
N/A	13	0	1	word-dep	0.8333
N/A	12	0	1	funcword	0.8333
N/A	10	0	1	word	0.8333
N/A	6	0	1	funcword	0.8333
N/A	5	0	1	pos-dep	0.8333
N/A	3	0	1	word	0.8333
spatial	44	0	2	pos-dep	0.7917
spatial	18	0	2	pos	0.7917
spatial	15	0	2	pos-dep	0.7917
spatial	14	0	2	pos	0.7917
spatial	14	0	2	pos-dep	0.7917
spatial	13	0	2	dep	0.7917
spatial	12	2	2	pos-dep	0.7917
spatial	12	0	2	pos	0.7917
spatial	10	3	2	pos-dep	0.7917
spatial	9	3	2	pos	0.7917
spatial	9	3	2	pos-dep	0.7917
spatial	9	2	2	pos	0.7917
spatial	9	2	2	pos-dep	0.7917
spatial	8	2	2	pos	0.7917
spatial	8	0	3	dep	0.7917
spatial	7	0	3	dep	0.7917
spatial	7	0	2	pos	0.7917
spatial	6	3	2	pos	0.7917
spatial	6	1	2	pos	0.7917
spatial	5	2	2	pos	0.7917
spatial	5	2	2	pos-dep	0.7917
spatial	5	0	3	dep	0.7917
spatial	4	3	2	pos	0.7917
spatial	4	2	2	pos-dep	0.7917
spatial	4	1	3	dep	0.7917
spatial	4	0	3	pos	0.7917

spatial	3	2	2	pos	0.7917
spatial	3	0	3	dep	0.7917
spatial	3	0	2	dep	0.7917
spatial	2	3	3	pos	0.7917
spatial	2	3	2	pos	0.7917
spatial	1	3	3	dep	0.7917
spatial	1	2	2	dep	0.7917
spatial	1	1	3	pos	0.7917
spatial	1	0	3	pos	0.7917
spatial	1	0	3	dep	0.7917
spatial	1	0	2	pos-dep	0.7917
N/A	59	0	1	funcword	0.7917
N/A	36	0	1	word	0.7917
N/A	32	0	1	funcword	0.7917
N/A	31	0	1	pos-dep	0.7917
N/A	30	0	1	funcword	0.7917
N/A	21	0	1	word-dep	0.7917
N/A	18	0	1	word-dep	0.7917
N/A	17	0	1	wordshape-word	0.7917
N/A	16	0	1	funcword	0.7917
N/A	13	0	1	wordshape-word	0.7917
N/A	13	0	1	funcword	0.7917
N/A	12	0	1	word	0.7917
N/A	12	0	1	wordshape-word	0.7917
N/A	12	0	1	word-dep	0.7917
N/A	11	0	1	word	0.7917
N/A	10	0	1	wordshape-word	0.7917
N/A	10	0	1	word-dep	0.7917
N/A	9	0	1	wordshape-word	0.7917
N/A	7	0	1	funcword	0.7917
N/A	6	0	1	word	0.7917
N/A	6	0	1	wordshape-word	0.7917
N/A	5	0	1	word	0.7917
N/A	2	0	1	wordshape-word	0.7917
N/A	2	0	1	word-dep	0.7917
N/A	1	0	1	wordshape-word	0.7917
N/A	1	0	1	pos-dep	0.7917
spatial	45	0	2	pos-dep	0.7500
spatial	37	3	2	pos-dep	0.7500
spatial	22	0	2	dep	0.7500
spatial	20	2	2	pos	0.7500
spatial	19	1	2	pos-dep	0.7500
spatial	19	0	2	dep	0.7500
spatial	18	3	2	pos	0.7500
spatial	17	0	2	dep	0.7500

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

spatial	15	1	2	pos	0.7500
spatial	15	0	2	pos	0.7500
spatial	14	3	2	pos	0.7500
spatial	14	2	2	pos	0.7500
spatial	14	0	2	dep	0.7500
spatial	13	3	2	pos	0.7500
spatial	13	3	2	pos-dep	0.7500
spatial	13	2	2	pos	0.7500
spatial	13	2	2	pos-dep	0.7500
spatial	13	1	2	pos	0.7500
spatial	13	0	3	dep	0.7500
spatial	13	0	2	pos-dep	0.7500
spatial	12	3	3	dep	0.7500
spatial	12	3	2	pos	0.7500
spatial	12	3	2	pos-dep	0.7500
spatial	12	0	2	pos-dep	0.7500
spatial	11	3	2	pos-dep	0.7500
spatial	11	2	2	pos	0.7500
spatial	11	2	2	pos-dep	0.7500
spatial	11	0	2	pos-dep	0.7500
spatial	11	0	2	dep	0.7500
spatial	10	3	2	pos	0.7500
spatial	10	2	2	pos	0.7500
spatial	10	0	2	pos	0.7500
spatial	10	0	2	pos-dep	0.7500
spatial	9	3	2	word-dep	0.7500
spatial	9	1	2	pos	0.7500
spatial	9	1	2	dep	0.7500
spatial	9	0	2	pos	0.7500
spatial	8	3	2	pos	0.7500
spatial	8	3	2	pos-dep	0.7500
spatial	8	1	2	pos	0.7500
spatial	8	1	2	pos-dep	0.7500
spatial	8	0	2	pos	0.7500
spatial	7	3	2	pos	0.7500
spatial	7	2	2	pos	0.7500
spatial	7	1	2	pos-dep	0.7500
spatial	6	3	2	pos-dep	0.7500
spatial	6	2	3	dep	0.7500
spatial	6	2	2	pos	0.7500
spatial	6	2	2	dep	0.7500
spatial	6	1	2	pos-dep	0.7500
spatial	6	0	3	dep	0.7500
spatial	6	0	2	pos-dep	0.7500
spatial	6	0	2	dep	0.7500

spatial	5	3	2	pos	0.7500
spatial	5	3	2	pos-dep	0.7500
spatial	5	1	3	dep	0.7500
spatial	5	1	2	dep	0.7500
spatial	4	0	2	pos-dep	0.7500
spatial	3	3	3	dep	0.7500
spatial	3	3	2	pos	0.7500
spatial	3	3	2	pos-dep	0.7500
spatial	3	1	3	pos	0.7500
spatial	3	1	2	pos-dep	0.7500
spatial	3	0	2	pos-dep	0.7500
spatial	2	3	2	pos-dep	0.7500
spatial	2	3	2	dep	0.7500
spatial	2	2	3	pos	0.7500
spatial	2	1	3	dep	0.7500
spatial	1	3	3	pos	0.7500
spatial	1	2	3	pos	0.7500
spatial	1	2	3	dep	0.7500
spatial	1	1	3	dep	0.7500
spatial	1	1	2	pos-dep	0.7500
spatial	1	1	2	dep	0.7500
N/A	58	0	1	funcword	0.7500
N/A	57	0	1	funcword	0.7500
N/A	56	0	1	funcword	0.7500
N/A	50	0	1	funcword	0.7500
N/A	49	0	1	word	0.7500
N/A	49	0	1	wordshape-word	0.7500
N/A	49	0	1	funcword	0.7500
N/A	48	0	1	funcword	0.7500
N/A	41	0	1	word	0.7500
N/A	37	0	1	word	0.7500
N/A	36	0	1	wordshape-word	0.7500
N/A	34	0	1	funcword	0.7500
N/A	32	0	1	pos-dep	0.7500
N/A	31	0	1	word-pos	0.7500
N/A	31	0	1	funcword	0.7500
N/A	30	0	1	word-pos	0.7500
N/A	28	0	1	word-pos	0.7500
N/A	28	0	1	funcword	0.7500
N/A	27	0	1	word-pos	0.7500
N/A	26	0	1	word-pos	0.7500
N/A	26	0	1	word-dep	0.7500
N/A	24	0	1	word-dep	0.7500
N/A	23	0	1	word-dep	0.7500
N/A	23	0	1	funcword	0.7500

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

N/A	22	0	1	word-dep	0.7500
N/A	20	0	1	funcword	0.7500
N/A	19	0	1	word-dep	0.7500
N/A	13	0	1	word-pos	0.7500
N/A	11	0	1	funcword	0.7500
N/A	10	0	1	funcword	0.7500
N/A	8	0	1	wordshape-word	0.7500
N/A	7	0	1	wordshape-word	0.7500
N/A	7	0	1	word-pos	0.7500
N/A	6	0	1	word-pos	0.7500
N/A	5	0	1	funcword	0.7500
N/A	4	0	1	word	0.7500
N/A	3	0	1	wordshape-word	0.7500
N/A	3	0	1	pos-dep	0.7500
N/A	3	0	1	funcword	0.7500
N/A	2	0	1	pos	0.7500
N/A	2	0	1	funcword	0.7500
spatial	58	3	3	wordshape	0.7083
spatial	57	3	3	wordshape	0.7083
spatial	53	2	2	wordshape	0.7083
spatial	34	3	3	wordshape	0.7083
spatial	31	0	2	dep	0.7083
spatial	30	2	2	dep	0.7083
spatial	29	2	2	dep	0.7083
spatial	28	0	2	dep	0.7083
spatial	25	1	2	dep	0.7083
spatial	25	0	2	dep	0.7083
spatial	24	3	2	word-dep	0.7083
spatial	24	0	2	dep	0.7083
spatial	23	0	2	dep	0.7083
spatial	21	3	2	funcword	0.7083
spatial	20	1	2	dep	0.7083
spatial	20	0	2	dep	0.7083
spatial	18	2	2	funcword	0.7083
spatial	18	0	2	dep	0.7083
spatial	17	3	3	dep	0.7083
spatial	17	2	3	dep	0.7083
spatial	16	3	3	dep	0.7083
spatial	15	0	3	dep	0.7083
spatial	15	0	2	dep	0.7083
spatial	14	3	2	word-pos	0.7083
spatial	13	3	3	pos-dep	0.7083
spatial	13	3	3	dep	0.7083
spatial	13	1	2	dep	0.7083
spatial	12	2	3	dep	0.7083

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

	spatial	12	2	2	dep	0.7083
	spatial	12	1	2	dep	0.7083
	spatial	12	0	2	word-pos	0.7083
	spatial	11	3	3	dep	0.7083
	spatial	11	1	3	pos-dep	0.7083
	spatial	10	2	2	word-dep	0.7083
	spatial	10	1	2	dep	0.7083
	spatial	10	0	3	dep	0.7083
	spatial	8	3	2	dep	0.7083
	spatial	8	1	2	dep	0.7083
	spatial	7	2	3	dep	0.7083
	spatial	7	0	2	dep	0.7083
	spatial	6	1	3	dep	0.7083
	spatial	6	1	2	dep	0.7083
	spatial	5	3	2	dep	0.7083
	spatial	5	2	3	dep	0.7083
	spatial	5	1	2	pos-dep	0.7083
	spatial	3	3	2	word-pos	0.7083
	spatial	3	1	2	dep	0.7083
	spatial	2	1	2	funcword	0.7083
	spatial	2	1	2	dep	0.7083
	spatial	1	1	2	funcword	0.7083
1742	spatial	1	0	2	funcword	0.7083
	N/A	57	0	1	wordshape-word	0.7083
	N/A	56	0	1	wordshape-word	0.7083
	N/A	55	0	1	funcword	0.7083
	N/A	54	0	1	wordshape-word	0.7083
	N/A	53	0	1	word	0.7083
	N/A	52	0	1	wordshape-word	0.7083
	N/A	50	0	1	word	0.7083
	N/A	50	0	1	wordshape-word	0.7083
	N/A	48	0	1	word	0.7083
	N/A	48	0	1	wordshape-word	0.7083
	N/A	47	0	1	word	0.7083
	N/A	47	0	1	wordshape-word	0.7083
	N/A	43	0	1	word-pos	0.7083
	N/A	32	0	1	word	0.7083
	N/A	30	0	1	word	0.7083
	N/A	30	0	1	word-dep	0.7083
	N/A	29	0	1	word	0.7083
	N/A	29	0	1	word-pos	0.7083
	N/A	27	0	1	word-dep	0.7083
	N/A	25	0	1	word-dep	0.7083
	N/A	24	0	1	word	0.7083
	N/A	21	0	1	funcword	0.7083

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

N/A	19	0	1	word-pos	0.7083
N/A	17	0	1	word-dep	0.7083
N/A	16	0	1	wordshape-word	0.7083
N/A	16	0	1	word-dep	0.7083
N/A	15	0	1	word	0.7083
N/A	15	0	1	wordshape-word	0.7083
N/A	14	0	1	word-dep	0.7083
N/A	14	0	1	funcword	0.7083
N/A	13	0	1	word	0.7083
N/A	12	0	1	word-pos	0.7083
N/A	10	0	1	pos-dep	0.7083
N/A	9	0	1	word	0.7083
N/A	9	0	1	word-dep	0.7083
N/A	9	0	1	funcword	0.7083
N/A	7	0	1	word	0.7083
N/A	7	0	1	word-dep	0.7083
N/A	5	0	1	wordshape-word	0.7083
N/A	4	0	1	wordshape-word	0.7083
N/A	4	0	1	funcword	0.7083
N/A	3	0	1	word-dep	0.7083
N/A	2	0	1	word-pos	0.7083
N/A	1	0	1	word-pos	0.7083
N/A	1	0	1	funcword	0.7083
spatial	30	0	2	wordlen	0.6667
spatial	29	0	3	dep	0.6667
spatial	28	0	2	wordlen	0.6667
spatial	23	3	2	funcword	0.6667
spatial	22	3	2	funcword	0.6667
spatial	20	3	2	funcword	0.6667
spatial	18	3	2	funcword	0.6667
spatial	12	3	2	funcword	0.6667
spatial	11	1	2	funcword	0.6667
spatial	10	1	2	funcword	0.6667
spatial	9	1	2	funcword	0.6667
spatial	4	3	2	funcword	0.6667
N/A	60	0	1	funcword	0.6667
N/A	54	0	1	word	0.6667
N/A	54	0	1	funcword	0.6667
N/A	53	0	1	wordshape-word	0.6667
N/A	53	0	1	funcword	0.6667
N/A	52	0	1	word	0.6667
N/A	52	0	1	funcword	0.6667
N/A	51	0	1	word	0.6667
N/A	39	0	1	word	0.6667
N/A	38	0	1	word	0.6667

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

N/A	38	0	1	funcword	0.6667
N/A	37	0	1	funcword	0.6667
N/A	35	0	1	funcword	0.6667
N/A	32	0	1	wordshape-word	0.6667
N/A	31	0	1	word	0.6667
N/A	31	0	1	wordshape-word	0.6667
N/A	29	0	1	wordshape-word	0.6667
N/A	29	0	1	funcword	0.6667
N/A	27	0	1	word	0.6667
N/A	27	0	1	wordshape-word	0.6667
N/A	27	0	1	funcword	0.6667
N/A	26	0	1	funcword	0.6667
N/A	25	0	1	funcword	0.6667
N/A	24	0	1	wordshape-word	0.6667
N/A	24	0	1	funcword	0.6667
N/A	23	0	1	word	0.6667
N/A	23	0	1	wordshape-word	0.6667
N/A	22	0	1	wordshape-word	0.6667
N/A	21	0	1	word	0.6667
N/A	17	0	1	word	0.6667
N/A	16	0	1	word	0.6667
N/A	11	0	1	wordshape-word	0.6667
N/A	8	0	1	word	0.6667
N/A	8	0	1	funcword	0.6667

Table A10. Cluster of feature extraction methods that their differences against the 30 most accurate method are not statistically significant ($p > 0.05$) (S1000 problems set).

Direction	l	k	n	Gram	Accuracy
N/A	2	0	1	word	0.8730
N/A	2	0	1	word-pos	0.8510
N/A	2	0	1	word-dep	0.8460
N/A	3	0	1	wordshape-word	0.8410
N/A	3	0	1	word	0.8350
spatial	2	3	2	pos	0.8320
spatial	2	2	2	pos	0.8310
N/A	3	0	1	word-pos	0.8250
spatial	2	1	2	pos	0.8160
N/A	3	0	1	word-dep	0.8140
N/A	4	0	1	wordshape-word	0.8120
N/A	4	0	1	word	0.8100
N/A	2	0	1	pos-dep	0.8100
spatial	2	0	2	pos	0.8060
N/A	2	0	1	pos	0.8040
spatial	3	3	2	pos	0.8030
spatial	2	3	2	pos-dep	0.8030
N/A	5	0	1	word	0.8020
N/A	4	0	1	word-pos	0.8000

spatial	2	3	3	pos	0.7990
spatial	3	2	2	pos	0.7910
N/A	5	0	1	word-pos	0.7910
spatial	2	0	2	dep	0.7900
N/A	3	0	1	pos	0.7880
spatial	4	3	2	pos	0.7870
spatial	2	2	2	pos-dep	0.7870
spatial	3	1	2	pos	0.7860
N/A	6	0	1	word-pos	0.7850
N/A	3	0	1	pos-dep	0.7830
spatial	3	3	2	dep	0.7790
spatial	2	3	2	dep	0.7790
spatial	2	2	3	pos	0.7790
spatial	2	2	2	dep	0.7790
spatial	2	1	2	pos-dep	0.7780
spatial	2	1	2	dep	0.7770
N/A	7	0	1	wordshape-word	0.7770
N/A	4	0	1	pos-dep	0.7760
spatial	2	3	2	word-pos	0.7750
N/A	5	0	1	pos-dep	0.7750
N/A	4	0	1	word-dep	0.7710
spatial	2	0	2	pos-dep	0.7700
spatial	4	1	2	pos	0.7690
spatial	3	3	2	pos-dep	0.7690
spatial	4	2	2	pos	0.7680
spatial	3	2	2	dep	0.7660
spatial	3	1	2	dep	0.7640
N/A	2	0	1	wordshape-word	0.7640
N/A	6	0	1	pos-dep	0.7630
spatial	5	3	2	pos	0.7610
spatial	3	3	3	pos	0.7610
spatial	2	3	2	word-dep	0.7610
spatial	3	0	2	pos	0.7600
spatial	2	2	2	word-pos	0.7600
spatial	2	2	2	word-dep	0.7590
spatial	2	1	3	pos	0.7580
spatial	2	3	3	dep	0.7570
N/A	4	0	1	pos	0.7560
spatial	4	3	2	dep	0.7530
spatial	3	0	2	dep	0.7530
spatial	2	2	3	dep	0.7520
spatial	2	1	2	word-pos	0.7520
N/A	5	0	1	word-dep	0.7520
N/A	4	0	1	dep	0.7500

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

Appendix B KIT-30 Per-author Statistics of Dutch, Greek, Spanish and US tweets

Tables A11, A12, A13 and A14 present the per-author statistics of Dutch, Greek, Spanish and US English languages, respectively. Similarly, the statistics of their chunked datasets are presented in Tables A15, A16, A17 and A18, respectively.

Table A11. KIT-30 statistics of the Dutch tweets subset.

Author ID	#Tweets	Avg. #letters/tweet	Avg. #words/tweet
NL-1	622	65.9566	10.7090
NL-2	2023	96.6960	14.9100
NL-3	820	71.2451	11.3256
NL-4	2907	67.3189	11.3261
NL-5	1244	80.4260	13.2267
NL-6	924	92.1926	14.4827
NL-7	2750	60.4389	10.1167
NL-8	962	78.1237	12.3981
NL-9	2077	87.8859	14.7525
NL-10	1797	56.8993	9.0117
NL-11	2655	89.6972	15.0580
NL-12	2665	71.1039	11.9343
NL-13	3025	60.8393	11.1504
NL-14	3161	60.3363	10.4960
NL-15	2761	55.7139	9.3115
NL-16	2799	40.0136	7.1111
NL-17	1394	86.5789	13.9326
NL-18	2540	91.1575	15.4394
NL-19	2067	83.3440	13.6420
NL-20	2872	82.8948	13.7256
NL-21	2971	84.6220	13.9458
NL-22	1210	82.2612	13.1645
NL-23	450	69.9844	11.2622
NL-24	2940	76.9707	13.5473
NL-25	2991	59.0779	10.4223
NL-26	1552	72.8640	12.1939
NL-27	2783	78.7298	13.6277
NL-28	1959	75.7264	13.1547
NL-29	908	79.6916	12.0220
NL-30	2189	79.7602	12.6944

Table A12. KIT-30 statistics of the Greek tweets subset.

Author ID	#Tweets	Avg. #letters/tweet	Avg. #words/tweet
GR-1	3205	94.2440	13.6003
GR-2	3220	93.7484	14.0823
GR-3	1635	83.4190	14.6000
GR-4	2409	57.1519	9.7007
GR-5	2467	67.1678	10.4929
GR-6	822	53.1569	8.6338
GR-7	2859	54.8723	8.9825

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

GR-8	2859	63.9185	10.6726
GR-9	423	84.8487	12.9314
GR-10	2984	60.5948	10.1555
GR-11	3093	59.8031	10.1856
GR-12	2851	42.0968	5.7825
GR-13	3146	69.3442	10.9784
GR-14	308	44.4026	6.8182
GR-15	3237	96.8119	14.8539
GR-16	998	61.2986	10.3878
GR-17	3171	55.6042	9.2122
GR-18	1917	67.0616	11.8623
GR-19	1772	98.5474	16.6986
GR-20	2103	84.4066	13.3538
GR-21	3091	87.0692	12.7299
GR-22	2037	68.5390	10.0520
GR-23	1905	79.7034	13.0084
GR-24	2361	82.1707	13.3943
GR-25	2960	62.4159	10.6885
GR-26	964	56.4378	8.8434
GR-27	397	73.0202	11.2720
GR-28	1193	108.0084	16.5541
GR-29	2212	62.9204	10.3250
GR-30	1023	86.2678	13.5181

Table A13. KIT-30 statistics of the Spanish tweets subset.

Author ID	#Tweets	Avg. #letters/tweet	Avg. #words/tweet
SP-1	2636	58.2159	9.0588
SP-2	1928	64.5799	10.2516
SP-3	2396	77.7496	12.8598
SP-4	1193	77.2372	13.5876
SP-5	2208	61.8750	9.8356
SP-6	609	74.3350	12.5057
SP-7	1453	53.7715	8.6084
SP-8	1369	91.0314	15.2579
SP-9	2622	56.2471	9.5393
SP-10	2101	65.7097	10.8920
SP-11	1023	83.5308	14.1496
SP-12	1496	84.6898	14.1203
SP-13	1279	85.2525	13.6013
SP-14	950	82.4411	14.0716
SP-15	2599	76.4833	12.2424
SP-16	2759	76.9880	12.7390
SP-17	2609	81.0356	13.4741
SP-18	1281	49.6979	7.3927
SP-19	2288	78.8383	13.1289
SP-20	2655	75.5156	11.9857
SP-21	2924	59.9863	10.3639

 doi:10.20944/preprints201708.0003.v1

SP-22	2253	64.3555	10.1336
SP-23	1963	48.4371	8.5074
SP-24	2960	66.9740	10.7426
SP-25	2771	57.7420	10.4543
SP-26	1651	68.6548	11.6893
SP-27	1498	85.8812	14.5294
SP-28	2390	100.9782	17.5971
SP-29	1350	69.3437	11.1689
SP-30	2782	61.4069	10.5201

Table A14. KIT-30 statistics of the US tweets subset.

Author ID	#Tweets	Avg. #letters/tweet	Avg. #words/tweet
US-1	3046	55.9409	10.2692
US-2	3202	77.1533	12.3998
US-3	2965	39.1696	7.3096
US-4	3057	72.4475	10.7053
US-5	1980	100.2116	18.4707
US-6	1300	57.9269	9.7377
US-7	3002	60.7921	10.7372
US-8	3109	52.1708	9.7327
US-9	2811	43.0893	6.1384
US-10	3225	85.2152	12.9460
US-11	2472	58.0546	10.3794
US-12	2946	71.7790	12.5350
US-13	1244	58.5314	11.2733
US-14	1091	24.6389	4.5215
US-15	1173	67.1816	12.1995
US-16	618	71.7799	12.9709
US-17	3095	43.8656	8.1958
US-18	2076	126.9282	20.2717
US-19	1668	63.9430	10.8064
US-20	2694	49.6533	9.3552
US-21	717	55.5565	9.2371
US-22	2076	79.4326	13.8516
US-23	2811	44.2686	7.9178
US-24	3205	49.3329	9.2256
US-25	2792	69.2110	12.6239
US-26	3067	41.3476	8.0492
US-27	2927	60.0215	10.2064
US-28	3005	52.6869	9.8093
US-29	1324	43.1193	8.0385
US-30	2139	51.6022	9.5283

Table A15. Chunked KIT-30 statistics of the Dutch tweets subset.

Author ID	#Chunks	Avg. #letters/chunk	Avg. #words/chunk
NL-1	10	4102.5	666.1
NL-2	10	19561.6	3016.3
NL-3	10	5842.1	928.7

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

1764

NL-4	10	19569.6	3292.5
NL-5	10	10005	1645.4
NL-6	10	8518.6	1338.2
NL-7	10	16620.7	2782.1
NL-8	10	7515.5	1192.7
NL-9	10	18253.9	3064.1
NL-10	10	10224.8	1619.4
NL-11	10	23814.6	3997.9
NL-12	10	18949.2	3180.5
NL-13	10	18403.9	3373
NL-14	10	19072.3	3317.8
NL-15	10	15382.6	2570.9
NL-16	10	11199.8	1990.4
NL-17	10	12069.1	1942.2
NL-18	10	23154	3921.6
NL-19	10	17227.2	2819.8
NL-20	10	23807.4	3942
NL-21	10	25141.2	4143.3
NL-22	10	9953.6	1592.9
NL-23	10	3149.3	506.8
NL-24	10	22629.4	3982.9
NL-25	10	17670.2	3117.3
NL-26	10	11308.5	1892.5
NL-27	10	21910.5	3792.6
NL-28	10	14834.8	2577
NL-29	10	7236	1091.6
NL-30	10	17459.5	2778.8

1765 **Table A16.** Chunked KIT-30 statistics of the Greek tweets subset.

1766

Author ID	#Chunks	Avg. #letters/chunk	Avg. #words/chunk
GR-1	10	30205.2	4358.9
GR-2	10	30187	4534.5
GR-3	10	13639	2387.1
GR-4	10	13767.9	2336.9
GR-5	10	16570.3	2588.6
GR-6	10	4369.5	709.7
GR-7	10	15688	2568.1
GR-8	10	18274.3	3051.3
GR-9	10	3589.1	547
GR-10	10	18081.5	3030.4
GR-11	10	18497.1	3150.4
GR-12	10	12001.8	1648.6
GR-13	10	21815.7	3453.8
GR-14	10	1367.6	210
GR-15	10	31338	4808.2
GR-16	10	6117.6	1036.7
GR-17	10	17632.1	2921.2
GR-18	10	12855.7	2274

 doi:10.20944/preprints201708.0003.v1

GR-19	10	17462.6	2959
GR-20	10	17750.7	2808.3
GR-21	10	26913.1	3934.8
GR-22	10	13961.4	2047.6
GR-23	10	15183.5	2478.1
GR-24	10	19400.5	3162.4
GR-25	10	18475.1	3163.8
GR-26	10	5440.6	852.5
GR-27	10	2898.9	447.5
GR-28	10	12885.4	1974.9
GR-29	10	13918	2283.9
GR-30	10	8825.2	1382.9

Table A17. Chunked KIT-30 statistics of the Spanish tweets subset.

Author ID	#Chunks	Avg. #letters/chunk	Avg. #words/chunk
SP-1	10	15345.7	2387.9
SP-2	10	12451	1976.5
SP-3	10	18628.8	3081.2
SP-4	10	9214.4	1621
SP-5	10	13662	2171.7
SP-6	10	4527	761.6
SP-7	10	7813	1250.8
SP-8	10	12462.2	2088.8
SP-9	10	14748	2501.2
SP-10	10	13805.6	2288.4
SP-11	10	8545.2	1447.5
SP-12	10	12669.6	2112.4
SP-13	10	10903.8	1739.6
SP-14	10	7831.9	1336.8
SP-15	10	19878	3181.8
SP-16	10	21241	3514.7
SP-17	10	21142.2	3515.4
SP-18	10	6366.3	947
SP-19	10	18038.2	3003.9
SP-20	10	20049.4	3182.2
SP-21	10	17540	3030.4
SP-22	10	14499.3	2283.1
SP-23	10	9508.2	1670
SP-24	10	19824.3	3179.8
SP-25	10	16000.3	2896.9
SP-26	10	11334.9	1929.9
SP-27	10	12865	2176.5
SP-28	10	24133.8	4205.7
SP-29	10	9361.4	1507.8
SP-30	10	17083.4	2926.7

Table A18. Chunked KIT-30 statistics of the US tweets subset.

Author ID	#Chunks	Avg. #letters/chunk	Avg. #words/chunk

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

US-1	10	17039.6	3128
US-2	10	24704.5	3970.4
US-3	10	11613.8	2167.3
US-4	10	22147.2	3272.6
US-5	10	19841.9	3657.2
US-6	10	7530.5	1265.9
US-7	10	18249.8	3223.3
US-8	10	16219.9	3025.9
US-9	10	12112.4	1725.5
US-10	10	27481.9	4175.1
US-11	10	14351.1	2565.8
US-12	10	21146.1	3692.8
US-13	10	7281.3	1402.4
US-14	10	2688.1	493.3
US-15	10	7880.4	1431
US-16	10	4436	801.6
US-17	10	13576.4	2536.6
US-18	10	26350.3	4208.4
US-19	10	10665.7	1802.5
US-20	10	13376.6	2520.3
US-21	10	3983.4	662.3
US-22	10	16490.2	2875.6
US-23	10	12443.9	2225.7
US-24	10	15811.2	2956.8
US-25	10	19323.7	3524.6
US-26	10	12681.3	2468.7
US-27	10	17568.3	2987.4
US-28	10	15832.4	2947.7
US-29	10	5709	1064.3
US-30	10	11037.7	2038.1

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2017 doi:10.20944/preprints201708.0003.v1

References

1. Fridman, L.; Weber, S.; Greenstadt, R.; Kam, M. Active Authentication on Mobile Devices via Stylometry, Application Usage, Web Browsing, and GPS Location. *IEEE Systems Journal* **2016**, *PP*, 1–9.
2. Pokhriyal, N.; Tayal, K.; Nwogu, I.; Govindaraju, V. Cognitive-Biometric Recognition From Language Usage: A Feasibility Study. *IEEE Transactions on Information Forensics and Security* **2017**, *12*, 134–143.
3. Fridman, L.; Stolerman, A.; Acharya, S.; Brennan, P.; Juola, P.; Greenstadt, R.; Kam, M. Multi-modal Decision Fusion for Continuous Authentication. *Comput. Electr. Eng.* **2015**, *41*, 142–156.
4. Igawa, R.A.; de Almeida, A.M.G.; Zarpelao, B.B.; Barbon, Jr, S. Recognition of Compromised Accounts on Twitter. Proceedings of the Annual Conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective - Volume 1; Brazilian Computer Society: Porto Alegre, Brazil, Brazil, 2015; SBSI 2015, pp. 2:9–2:14.
5. Yang, Z.; Davison, B.D. Venue Recommendation: Submitting Your Paper with Style. 2012 11th International Conference on Machine Learning and Applications, 2012, Vol. 1, pp. 681–686.
6. Daelemans, W. Explanation in Computational Stylometry. Proceedings of the 14th CICLing; Springer-Verlag: Berlin, Heidelberg, 2013; Vol. 2, pp. 451–462.
7. Caliskan-Islam, A.; Harang, R.; Liu, A.; Narayanan, A.; Voss, C.; Yamaguchi, F.; Greenstadt, R. De-anonymizing Programmers via Code Stylometry. 24th USENIX Security Symposium (USENIX Security 15); USENIX Association: Washington, D.C., 2015; pp. 255–270.
8. Islam, A.C.; Yamaguchi, F.; Dauber, E.; Harang, R.E.; Rieck, K.; Greenstadt, R.; Narayanan, A. When Coding Style Survives Compilation: De-anonymizing Programmers from Executable Binaries. *CoRR* **2015**, *abs/1512.08546*.
9. Koppel, M.; Schler, J.; Bonchek-Dokow, E. Measuring Differentiability: Unmasking Pseudonymous Authors. *Journal of Machine Learning Research* **2007**, *8*, 1261–1276.
10. Juola, P. Authorship Attribution. *Foundations and Trends in Information Retrieval.* **2006**, *1*, 233–334.
11. Koppel, M.; Schler, J.; Argamon, S. Computational Methods in Authorship Attribution. *J. Am. Soc. Inf. Sci. Technol.* **2009**, *60*, 9–26.
12. Stamatatos, E. A Survey of Modern Authorship Attribution Methods. *J. Am. Soc. Inf. Sci. Technol.* **2009**, *60*, 538–556.
13. Joula, P.; Stamatatos, E. Overfiew of the Author Identification Task at PAN 2013. Conference and Labs of the Evaluation Forum, 2013.
14. Stamatatos, E.; Daelemans, W.; Verhoeven, B.; Stein, B.; Potthast, M.; Juola, P.; Sánchez-Pérez, M.A.; Barrón-Cedeño, A. Overview of the Author Identification Task at PAN 2014. CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers; , 2014.
15. Stamatatos, E.; amd Ben Verhoeven, W.D.; Juola, P.; López-López, A.; Potthast, M.; Stein, B. Overview of the Author Identification Task at PAN 2015. CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France; Cappellato, L.; Ferro, N.; Jones, G.; San Juan, E., Eds. CEUR-WS.org, 2015.
16. Yule, G.U. The Statistical Study of Literary Vocabulary. *Cambridge University Press* **1944**.
17. Honore, A. Some Simple Measure of Richness of Vocabulary. *Association for Literary and Linguistic Computing Bulletin* **1979**, pp. 172–177.
18. Tanaka-Ishii, K.; Aihara, S. Computational Constancy Measures of Texts-yule's K and RéNyi's Entropy. *Comput. Linguist.* **2015**, *41*, 481–502.
19. Sidorov, G.; Velasquez, F.; Stamatatos, E.; Gelbukh, A.; Chanona-Hernández, L., Advances in Computational Intelligence; Springer Berlin Heidelberg: Berlin, Heidelberg, 2013; chapter Syntactic Dependency-Based N-grams as Classification Features, pp. 1–11.
20. Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; Khudanpur, S. Recurrent Neural Network Based Language Model. 11th Annual Conference of the International Speech Communication Association, 2010, pp. 1045–1048.
21. Bagnall, D. Author Identification using multi-headed Recurrent Neural Networks. CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France; Cappellato, L.; Ferro, N.; Jones, G.; San Juan, E., Eds. CEUR-WS.org, 2015.

22. Eder, M.; Rybicki, J.; Kestemont, M. Stylometry with R: A Package for Computational Text Analysis. *The R Journal* **2016**, *8*, 107–121.
23. Guthrie, D.; Allison, B.; Liu, W.; Guthrie, L.; Wilks, Y. A Closer Look at Skip-Gram Modelling. Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006); , 2006.
24. Mikros, G.K.; Argiri, E.K. Investigating Topic Influence in Authorship Attribution. PAN; Stein, B.; Koppel, M.; Stamatatos, E., Eds., 2007, Vol. 276, *CEUR Workshop Proceedings*.
25. Kestemont, M.; Luyckx, K.; Daelemans, W.; Crombez, T. Cross-Genre Authorship Verification Using Unmasking. 2012, Vol. 93, *English Studies*, pp. 340–356.
26. Sapkota, U.; Solorio, T.; Montes, M.; Bethard, S.; Rosso, P. Cross-Topic Authorship Attribution: Will Out-Of-Topic Data Help? Proceedings of COLING 2014. ACL, 2014, pp. 1228–1237.
27. Schwartz, R.; Tsur, O.; Rappoport, A.; Koppel, M. Authorship Attribution of Micro-Messages. EMNLP. ACL, 2013, pp. 1880–1891.
28. Brennan, M.; Afroz, S.; Greenstadt, R. Adversarial Stylometry: Circumventing Authorship Recognition to Preserve Privacy and Anonymity. *ACM Trans. Inf. Syst. Secur.* **2012**, *15*, 12:1–12:22.
29. Khonji, M.; Iraqi, Y. Stylometric Anonymity: Is Imitation the Best Strategy? IEEE Trustcom/BigDataSE/ISPA, 2015, Vol. 1, pp. 974–982.
30. de Vel, O.; Anderson, A.; Corney, M.; Mohay, G. Mining e-Mail Content for Author Identification Forensics. *SIGMOD Rec.* **2001**, *30*, 55–64.
31. Diederich, J.; Kindermann, J.; Leopold, E.; Paass, G. Authorship Attribution with Support Vector Machines. *Applied Intelligence*, *19*, 109–123.
32. fa Teng, G.; sheng Lai, M.; Ma, J.B.; Li, Y. E-mail authorship mining based on SVM for computer forensic. Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, 2004, Vol. 2, pp. 1204–1207.
33. Li, J.; Zheng, R.; Chen, H. From Fingerprint to Writeprint. *Commun. ACM* **2006**, *49*, 76–82.
34. Sanderson, C.; Guenter, S. Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: An Investigation. EMNLP. ACL, 2006, pp. 482–491.
35. Uzuner, O.; Katz, B. A Comparative Study of Language Models for Book and Author Recognition. Proceedings of the Second International Joint Conference on Natural Language Processing; Springer-Verlag: Berlin, Heidelberg, 2005; pp. 969–980.
36. Zhao, Y.; Zobel, J. Effective and Scalable Authorship Attribution Using Function Words. Proceedings of the Second Asia Conference on Asia Information Retrieval Technology, 2005, pp. 174–189.
37. Zheng, R.; Li, J.; Chen, H.; Huang, Z. A Framework for Authorship Identification of Online Messages: Writing-style Features and Classification Techniques. *J. Am. Soc. Inf. Sci. Technol.* **2006**, *57*, 378–393.
38. Khonji, M.; Iraqi, Y.; Jones, A. An Evaluation of Authorship Attribution Using Random Forests. Information and Communication Technology Research (ICTRC), 2015 International Conference on, 2015, pp. 68–71.
39. Fréry, J.; Largeron, C.; Juganaru-Mathieu, M. UJM at CLEF in Author Identification. CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers, 15-18 September, Sheffield, UK; Cappellato, L.; Ferro, N.; Halvey, M.; Kraaij, W., Eds. CEUR-WS.org, 2014.
40. MATTHEWS, R.A.J.; MERRIAM, T.V.N. Neural Computation in Stylometry I: An Application to the Works of Shakespeare and Fletcher. *Literary and Linguistic Computing* **1993**, *8*, 203–209.
41. MERRIAM, T.V.N.; MATTHEWS, R.A.J. Neural Computation in Stylometry II: An Application to the Works of Shakespeare and Marlowe. *Literary and Linguistic Computing* **1994**, *9*, 1–6.
42. Tweedie, F.J.; Singh, S.; Holmes, D.I. Neural network applications in stylometry: The Federalist Papers. *Computers and the Humanities*, *30*, 1–10.
43. Khosmood, F.; Levinson, R. Toward Unification of Source Attribution Processes and Techniques. Machine Learning and Cybernetics, 2006 International Conference on, 2006, pp. 4551–4556.
44. Stamatatos, E.; Kokkinakis, G.; Fakotakis, N. Automatic Text Categorization in Terms of Genre and Author. *Comput. Linguist.* **2000**, *26*, 471–495.
45. Tambouratzis, G.; Markantonatou, S.; Hairetakis, N.; Vassiliou, M.; Tambouratzis, D.; Carayannis, G. Discriminating the Registers and Styles in the Modern Greek Language. Proceedings of the Workshop on Comparing Corpora - Volume 9; ACL: Stroudsburg, PA, USA, 2000; pp. 35–42.
46. Chaski, C.E. Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations. *IJDE* **2005**, *4*.

47. Luyckx, K.; Daelemans, W., Shallow Text Analysis and Machine Learning for Authorship Attribution. In *Proceedings of the 15th meeting of Computational Linguistics in the Netherlands (CLIN 2004)*; 2005; pp. 149–160.
48. Mosteller, F.; Wallace, D.L. *Inference and Disputed Authorship: The Federalist Papers*; Addison-Wesley: Reading, Mass., 1964.
49. Clement, Ross and Sharp, David. Ngram and Bayesian Classification of Documents for Topic and Authorship. *Literary and Linguistic Computing* **2003**, *18*, 423–447.
50. Peng, F.; Schuurmans, D.; Wang, S. Augmenting Naive Bayes Classifiers with Statistical Language Models. *Inf. Retr.* **2004**, *7*, 317–345.
51. Zhao, Y.; Zobel, J., Information Retrieval Technology: Second Asia Information Retrieval Symposium; Springer Berlin Heidelberg: Berlin, Heidelberg, 2005; chapter Effective and Scalable Authorship Attribution Using Function Words, pp. 174–189.
52. Keselj, V.; Peng, F.; Cercone, N.; Thomas, C. N-gram-based Author Profiles for Authorship Attribution. Proceedings of PACL. PACL, 2003, pp. 255–264.
53. Frantzeskou, G.; Stamatatos, E.; Gritzalis, S.; Katsikas, S., Artificial Intelligence Applications and Innovations: 3rd IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI) 2006, June 7–9, 2006, Athens, Greece; Springer US: Boston, MA, 2006; chapter Source Code Author Identification Based on N-gram Author Profiles, pp. 508–515.
54. Jankowska, M.; Milios, E.; Keselj, V. Author Verification using Common N-gram Profiles of Text Documents. Proceedings of COLING 2014, 2014, pp. 387–397.
55. Keogh, E.; Lonardi, S.; Ratanamahatana, C.A. Towards Parameter-free Data Mining. Proceedings of the Tenth ACM SIGKDD; ACM: New York, NY, USA, 2004; pp. 206–215.
56. Cilibrasi, R.; Vitányi, P.M.B. Clustering by compression. *IEEE Transactions on Information Theory* **2005**, *51*, 1523–1545.
57. de Graaff, R.; Veenman, C.J. Bootstrapped Authorship Attribution in Compression Space. Notebook for PAN at CLEF, 2012.
58. Veenman, C.; Li, Z. Authorship Verification with Compression Features. CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain; Forner, P.; Navigli, R.; Tufis, D., Eds., 2013.
59. Burrows, J. 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing* **2002**, *17*, 267–287.
60. Argamon, S. Interpreting Burrows's Delta: Geometric and Probabilistic Foundations. *Literary and Linguistic Computing* **2008**, *23*, 131–147.
61. Jannidis, F.; Pielström, S.; Schöch, C.; Vitt, T. Improving Burrows' Delta – An empirical evaluation of text distance measures. Book of Abstracts of the Digital Humanities Conference 2015. ADHO, UWS, 2015.
62. Sanderson, C.; Guenter, S. Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: An Investigation. EMNLP; ACL: Stroudsburg, PA, USA, 2006; pp. 482–491.
63. Koppel, M.; Seidman, S. Automatically Identifying Pseudepigraphic Texts. EMNLP. ACL, 2013, pp. 1449–1454.
64. Khonji, M.; Iraqi, Y. A Slightly-modified GI-based Author-verifier with Lots of Features (ASGALF). CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers, 15-18 September, Sheffield, UK; Cappellato, L.; Ferro, N.; Halvey, M.; Kraaij, W., Eds. CEUR-WS.org, 2014.
65. Seroussi, Y.; Zukerman, I.; Bohnert, F. Authorship Attribution with Latent Dirichlet Allocation. Proceedings of the Fifteenth CoNLL; ACL: Stroudsburg, PA, USA, 2011; pp. 181–189.
66. Juola, P. An Overview of the Traditional Authorship Attribution Subtask. CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy; Forner, P.; Karlgren, J.; Womser-Hacker, C., Eds., 2012.
67. Nabil, M.; Aly, M.A.; Atiya, A.F. ASTD: Arabic Sentiment Tweets Dataset. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015; Màrquez, L.; Callison-Burch, C.; Su, J.; Pighin, D.; Marton, Y., Eds. The Association for Computational Linguistics, 2015, pp. 2515–2519.
68. Rocha, A.; Scheirer, W.J.; Forstall, C.W.; Cavalcante, T.; Theophilo, A.; Shen, B.; Carvalho, A.R.B.; Stamatatos, E. Authorship Attribution for Social Media Forensics. *IEEE Transactions on Information Forensics and Security* **2017**, *12*, 5–33.