SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining

Benjamin Weggenmann
SAP Security Research
benjamin.weggenmann@sap.com

Florian Kerschbaum
University of Waterloo
florian.kerschbaum@uwaterloo.ca

ABSTRACT

Text mining and information retrieval techniques have been developed to assist us with analyzing, organizing and retrieving documents with the help of computers. In many cases, it is desirable that the authors of such documents remain anonymous: Search logs can reveal sensitive details about a user, critical articles or messages about a company or government might have severe or fatal consequences for a critic, and negative feedback in customer surveys might negatively impact business relations if they are identified. Simply removing personally identifying information from a document is, however, insufficient to protect the writer's identity: Given some reference texts of suspect authors, so-called *authorship attribution* methods can reidentfy the author from the text itself.

One of the most prominent models to represent documents in many common text mining and information retrieval tasks is the vector space model where each document is represented as a vector, typically containing its term frequencies or related quantities. We therefore propose an automated text anonymization approach that produces synthetic term frequency vectors for the input documents that can be used in lieu of the original vectors. We evaluate our method on an exemplary text classification task and demonstrate that it only has a low impact on its accuracy. In contrast, we show that our method strongly affects authorship attribution techniques to the level that they become infeasible with a much stronger decline in accuracy. Other than previous authorship obfuscation methods, our approach is the first that fulfills *differential privacy* and hence comes with a provable plausible deniability guarantee.

CCS CONCEPTS

• **Security and privacy** → **Data anonymization and sanitization**; • **Information systems** → *Document representation*; **Clustering and classification**; Data mining;

KEYWORDS

Text Classification; Differential Privacy; Synthetic Data; Authorship Attribution; Authorship Obfuscation; Anonymization; Text Mining

ACM Reference Format:
Benjamin Weggenmann and Florian Kerschbaum. 2018. SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210008

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

1 INTRODUCTION

For centuries, text has been used to convey information between human beings through books, letters, newspapers and magazines. With the advent of the digital age, more and more textual data is being processed and analyzed by machines. Typical tasks include text classification, which is used in particular for spam filtering [36] and automated email routing [6], document retrieval [38], where indexed documents are retrieved and ranked according to search queries, sentiment analysis [23], and a wide variety of other tasks in the information retrieval (IR) and text mining domains.

In many cases, it is desirable for an author that his writings stay anonymous. This could be the case if the textual data contains sensitive information about the author, for instance in search queries. Negative feedback from customer surveys might negatively impact business relations if the author or his company is known, and critical news or blog articles about a company (or government) might have severe (or fatal) consequences for the author of the article. In other areas, anonymity is required for compliance or legal reasons, e.g. in the selection of job candidates to eliminate discrimination. Furthermore, without anonymity people and data owners might feel reluctant to participate in surveys or to release their data. Offering anonymity might be a means to convince them to share their data in an anonymized form, which could then be used to perform evaluations and as training data for machine learning models.

Traditional sanitization approaches for free text include removing parts containing personally identifiable information (PII) such as the author's name, or replacing it with a pseudonym. However, these methods are insufficient to protect the author's identity: As the famous Netflix de-anonymization attack [31] and other studies [9, 16, 35, 42] have shown, the *originator* of data can be *re-identified from the data itself.* We illustrate this in the case of the AOL search data release [5], where search queries of over 650,000 users were released for research purposes in 2006. The search logs were "anonymized" by linking the queries to a numerical identifier instead of the actual user name. After some investigation in the data, the New York Times eventually learned enough information about user 4417749 so they could re-identify her as Thelma Arnold, a 62-year-old widow from Lilburn, a city in Georgia.

The task of attributing authorship of an anonymous or disputed document to its respective author is called *authorship attribution*. Such methods usually make use of stylistic features to identify or discriminate authors, as has been done with the statistic techniques in [30] to resolve the dispute of the Federalist Papers. Recently, more sophisticated methods have evolved that use statistical analysis and machine learning to tackle the problem. A survey of modern authorship attribution methods is given by Stamatatos [40], and examples include the JGAAP [18] and JStylo [26] frameworks. While these powerful methods are useful in the literary world and in forensics, they can often pose a threat to the privacy and integrity of authors of documents with potentially sensitive content.

Contributions. Many IR and text mining algorithms rely on the vector space model (VSM) [38] where documents are represented as vectors containing, for instance, their term frequency (tf) or term frequency–inverse document frequency (tf–idf) values. Therefore, we propose a solution that targets this representation and produces synthetic tf vectors which can be used as a substitute for the original ones. More precisely, we make the following contributions:

- In section 3, we propose "SynTF", a differentially private method to compute anonymized, *synthetic term frequency vectors* for textual data that can be used as feature vectors for common IR and text mining tasks such as text classification.
- In section 3.4, we give theoretical results on the differential privacy properties of our method. We derive improved bounds for the privacy loss of our method and give a heuristic argument that differential privacy on large (discrete) output spaces demands a large privacy loss if the result should fulfill a minimum usefulness requirement.
- In section 4, we experimentally verify our method on a corpus of newsgroups postings: A benign, well-intended analyst wants to classify the documents into certain topics, whereas a malicious attacker tries to re-identify the author of these documents using authorship attribution techniques. The results show that our method has a much stronger impact on the attacker's than on the analyst's task.

Based on our motivation and results, we presume that the synthetic term frequency vectors (SynTF vectors) can be used in a multitude of text mining and IR tasks where the semantic similarity of documents is decisive. On the other hand, our method obliterates stylistic features that could otherwise reveal the identity and other privacy-sensitive information about the writer such as age or gender.

2 PRELIMINARIES

In this section, we briefly describe text classification, and follow with a more detailed introduction on differential privacy.

2.1 Text Classification

Text classification is the problem of assigning a given text to one or more predefined categories. It has many applications, for instance in the automated sorting and filtering of email messages, spam filtering, categorization of news articles, etc. The problem is typically solved using machine learning techniques. In the supervised model, a classifier is *trained* based on a set of documents with known categories so it can recognize characteristic features in the text that indicate the right category. A trained classifier can then *predict* the most likely category for new texts whose category is unknown.

To obtain a representation corresponding to the vector space and Bag-of-Words (BoW) models, documents are transformed into feature vectors where each entry corresponds to a certain word in an underlying vocabulary. The process of this transformation is also called *vectorization*. Two common representations are *term frequency* (tf) vectors where each entry equals the number of occurrences of the corresponding term in the document, and the *term frequency – inverse document frequency* (tf–idf) vectors which are derived from the tf vectors by also taking the number of documents into account that contain the corresponding term. We refer to [39] for more information on text mining and information retrieval.

2.2 Differential Privacy

Differential privacy has first been proposed by Dwork et al. [11] under the name *ϵ-indistinguishability*. It works by releasing noisy answers to the database queries, where the noisy results on two databases that differ in only a single record are probabilistically *indistinguishable* up to a multiplicative factor. We give some basic terminology and results as required in the paper. For a broader introduction and further details on differential privacy, we refer the reader to the book by Dwork and Roth [12]. We follow the notation of [8], with the one deviation that we describe random mechanisms via random variables instead of probability measures on the output space. Since every random variable induces a probability measure on the underlying space, the two definitions are equivalent.

Definition 2.1 (Randomized mechanism). Let $\mathcal{X}$ and $\mathcal{Z}$ be two sets where $\mathcal{Z}$ is measurable, and let $\mathcal{R}(\mathcal{Z})$ be the set of random variables on $\mathcal{Z}$. A *randomized mechanism* from $\mathcal{X}$ to $\mathcal{Z}$ is a probabilistic function $\mathcal{M} : \mathcal{X} \to \mathcal{R}(\mathcal{Z})$ that assigns a random variable on $\mathcal{Z}$ to each input $x \in \mathcal{X}$. From an algorithmic point of view, we *run* an instance of a randomized mechanism $\mathcal{M}$ on a given input x by *sampling* a realization z of the random variable $\mathcal{M}(x)$. We write this as $z \leftarrow_{\mathrm{R}} \mathcal{M}(x)$.

As noted above, each random variable on $\mathcal{Z}$ induces a probability distribution on $\mathcal{Z}$. A continuous/discrete distribution is typically described by its probability density/mass function (pdf/pmf). By slight abuse of notation, we write $\Pr[X = x]$ for the pdf/pmf of X.

Definition 2.2 (Adjacency). Given a metric $\mathrm{d}_{\mathcal{X}}$ on the space $\mathcal{X}$, we say that two inputs $x_1, x_2 \in \mathcal{X}$ are *adjacent (with respect to* $\mathrm{d}_{\mathcal{X}}$*)* if $\mathrm{d}_{\mathcal{X}}(x_1, x_2) \leq 1$. We write this as $x_1 \sim_{\mathrm{d}_{\mathcal{X}}} x_2$ (or $x_1 \sim x_2$ if the metric is unambiguous).

Definition 2.3 (Differential Privacy). Let $\epsilon > 0$ be a privacy parameter. A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{R}(\mathcal{Z})$ fullfils ϵ-*differential privacy* if for any two adjacent inputs $x_1, x_2 \in \mathcal{X}$, and any set of possible outputs $Z \subseteq \mathrm{Im}(\mathcal{M})$,

$$\Pr[\mathcal{M}(x_1) \in Z] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(x_2) \in Z].$$

The *privacy loss* of a randomized mechanism $\mathcal{M}$ is the quantity

$$\ell(\mathcal{M}) := \sup_{x_1 \sim x_2} \sup_{Z \in \mathrm{Im}(\mathcal{M})} \ln \frac{\Pr[\mathcal{M}(x_1) \in Z]}{\Pr[\mathcal{M}(x_2) \in Z]}.$$

Note that ϵ is an upper bound for the privacy loss, and hence any randomized mechanism $\mathcal{M}$ with finite privacy loss $\ell(\mathcal{M})$ also fulfills ϵ-differential privacy with $\epsilon = \ell(\mathcal{M})$.

Typically, the input space $\mathcal{X}$ models the set of databases over some domain of values $\mathcal{V}$ with n records, i.e. $\mathcal{X} = \mathcal{V}^n$. In the case of textual documents, we adopt the vector space/BoW model where each document x is represented as feature vector over some vocabulary V of size L. Since we anonymize each document independently, we assume $\mathcal{X} = \mathcal{Z} = \mathbb{R}^L_{\geq 0}$. We consider any two texts as adjacent which is the most strict and conservative way to define adjacency.

The Exponential Mechanism. A very important and versatile building block for differential privacy is the Exponential mechanism by McSherry and Talwar [27]. It applies to both numerical and categorical data and fulfills ϵ-differential privacy as shown in [27, theorem 6]. It requires a "measure of suitability" for each possible pair of inputs and outputs:

Definition 2.4 (Rating function and sensitivity). A function $\rho : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ is called a *rating function* from $\mathcal{X}$ to $\mathcal{Z}$. The value $\rho(x, z)$ is the *rating* for input x and output z. The *sensitivity* Δ_ρ of the rating function ρ is its largest possible difference given two adjacent inputs, over all possible output values:

$$\Delta_\rho := \max_{z \in \mathcal{Z}} \max_{x_1 \sim x_2} \big(\rho(x_1, z) - \rho(x_2, z)\big)$$

In our scenario with textual data, the rating function ρ will be bounded to $[0, 1]$, which implies that its sensitivity is $\Delta_\rho \leq 1$.

Definition 2.5 (Exponential mechanism). Let $\epsilon > 0$ be a privacy parameter, and let $\rho : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ be a rating function. For each $x \in \mathcal{X}$, we define a random variable $\mathcal{E}_{\epsilon,\rho}(x)$ that is described by the probability density function (pdf)

$$\Pr\left[\mathcal{E}_{\epsilon,\rho}(x) = z\right] = \frac{\exp\left(\frac{\epsilon}{2\Delta}\rho(x, z)\right)}{\int_{z'} \exp\left(\frac{\epsilon}{2\Delta}\rho(x, z')\right) dz'}.$$

Note that a discrete version of the Exponential mechanism for countable $\mathcal{Z}$ is obtained by replacing the integral with a sum.

3 SYNTHETIC TERM FREQUENCY VECTORS

In this section, we first describe the intended usage scenario. We then take a closer look under the hood of authorship attribution techniques and derive the basic motivation behind our SynTF method. Finally, we describe our method in detail and present its differential privacy properties.

3.1 Usage Scenario

Consider a data processor that wishes to share sensitive training data for machine learning with a third-party analyst. Feature vectors are sufficient for most machine learning tasks since they are produced by the analyst in a preprocessing step anyway. Our method automatically creates anonymized feature vectors that can be shared with the analyst and which he can use in lieu of his own vectors.

In our present scenario, we are given a set of text documents such as email messages, job applications or survey results. The documents shall be analyzed by a (benign) third-party analyst, who wants to perform a typical text mining task such as text classification. Our aim is to prevent authorship attribution attacks as described above. Therefore, to protect the identity of the authors and prevent re-identification, we only provide the analyst with synthetic BoW feature vectors instead of the original documents. Email providers and search engines could share anonymized feature vectors of emails or (aggregted) search queries with advertising networks to provide personalized ads while protecting their users.

Attacker Model. The attacker is presented with a document of unknown authorship which has been written by one of several suspected authors. Her goal is to identify the document's actual author from the group of suspects. We assume that she has a set of similar reference documents from each suspect that she can use to help decide which suspect to assign the unknown document to.

We compare the attacker's capability to re-identify the authors on the original plaintexts as well as the anonymized feature vectors. We assume the attacker knows the dictionary, so she can convert the numbers in the feature vectors to a textual representation by repeating each word accordingly. This allows her to (partially) deduce more complex features beyond BoW, such as the WritePrints feature set which is often used in authorship attribution [1, 26]. As explained in the next section 3.2, most of these features cannot be correctly inferred anymore, which is beneficial for our method as these are precisely the stylistic features (beyond BoW) that are exclusively exploited by our attacker.

3.2 Preventing Authorship Attribution

A popular feature set for authorship attribution has been described in the WritePrints method [1]. It includes the following types of stylistic features:

- **Lexical** Counts of letters, digits, special characters, number of characters and words, etc.
- **Syntactic** Frequency of function words, punctuation, parts of speech (POS) tags.
- **Structural** Number and length of paragraphs and sentences, URLs or quoted content, etc.
- **Content** Frequencies of words (BoW model).
- **Idiosyncratic** Misspelled words.

For some features such as letters, words, digits and POS tags, it also considers their bi- and trigrams, thus taking *order information* into account. These features have a strong capability to capture individual stylistic characteristics expressed by the writer of a text. For instance, one author might subconsciously prefer using the passive voice or past tense, so many verbs will end in an "ed"-bigram, whereas another author might tend to use the present continuous or gerund which causes many "ing"-trigrams.

Ordinary text mining and IR tasks such as classification typically only use content-level features which are often modeled and represented as term frequency vectors (tf vectors) in the BoW model. Most of the stylistic features used for authorship attribution thus get *lost in vectorization*: In fact, the tf *vectors* by their very nature do not capture any structural information, and most syntactic features will be destroyed as well. Apart from the content (and idiosyncratic) features, however, we can still derive lexical features if the BoW vocabulary is known.

Since the attacker can still exploit the derived lexical features, we aim at disturbing them in a way that keeps the meaning or theme of a document intact, thus further allowing the classification task but impairing authorship attribution. Lexical features are mostly related with the spelling, therefore, our idea is to replace words in the input with words with similar meaning (synonyms) but different spelling to make the lexical features meaningless for the attacker. On the other hand, this will preserve the general theme of the text, so we hope that the impact is little on the classification task.

3.3 The SynTF Mechanism

Our goal is a differentially private anonymization method to derive synthetic feature vectors that keeps the theme of the represented document intact and at the same time prevents authorship attribution attacks. For performance and memory efficiency reasons, we require our method to preserve the sparseness in the tf vectors. Simply applying Laplace noise [11] or differentially private histogram publication methods [47] will fail this requirement, since they produce dense vectors. Our core idea is to take a word count entry for one term in the tf vector and probabilistically distribute it across all terms in the pre-defined vocabulary. The probability of each term is determined according to its *similarity* with the original word. Word similarity can be expressed in various ways, cf. section 4.1.2.

Differential privacy presents a strong requirement for the method: Namely, *every* possible output must occur with non-zero possibility for *any* other input. This means that a statement on food preference can be processed to the same output as a conversation on politics, with non-zero probability. This has two implications: First, we must ensure that the probability of picking a term is always greater than zero, even for totally unrelated words. Second, it must be possible that two input texts of different lengths produce the same number of words in their resulting tf vectors. Therefore, we must also specify the output length. Note that this approach limits the number of entries that are changed from the original to the anonymized tf vector, so it keeps the sparseness of the resulting vector intact.

Algorithm Description. In the following, let V denote the underlying vocabulary of size $|V| = L$. The vocabulary could be derived, for instance, from a reference corpus of documents from a similar context as the target documents which shall be anonymized. We will describe the SynTF approach for a single document T, but it is possible to anonymize an entire corpus simultaneously. The anonymization for a document T consists of two main phases:

- **Analysis** We *vectorize* the document T to its feature vector $\mathbf{t} = (t_1, \dots, t_K) \in \mathbb{R}^K_{\geq 0}$. Typically, $\mathbf{t}$ will be a tf or tf–idf vector over the underlying vocabulary V. Next, we *normalize* $\mathbf{t}$ with respect to the ℓ_1-norm to transform it into a *composition vector* $\theta_\mathbf{t} := \mathbf{t}/\|\mathbf{t}\|_1$ whose entries can be interpreted as probability distribution over V.
- **Synthesis** We repeatedly sample terms $v_1, \dots, v_n$ from the distribution $\theta_\mathbf{t}$ on V. For each v_i, we use the Exponential mechanism to pick a substitute output term $w_i \in V$ with probability proportional to a *similarity rating* $\rho(v_i, w_i)$. Finally, we construct a synthetic tf vector $\mathbf{s} \in \mathbb{N}^L_{\geq 0}$ of length n by counting all the terms w_i.

Algorithm 1 illustrates the synthesis phase of our SynTF mechanism in pseudocode. It uses the following definition:

Definition 3.1 (Categorical distribution). For an enumerable set $V = \{v_1, \dots, v_k\}$ and associated probability vector $\mathbf{p} = (p_v)_{v \in V}$ with $\sum_{v \in V} p_v = 1$, the *categorical distribution*, denoted $\mathrm{Cat}(\mathbf{p})$, is defined on V through $\Pr[\mathrm{Cat}(\mathbf{p}) = v_i] = p_i$.

3.4 Differential Privacy Results

In this section, we give differential privacy-related results on our SynTF mechanism. We provide an extended technical report with full proofs in [46].

Algorithm 1: SynTF Term-Frequency Vector Synthesis

Input: document vector $\theta_\mathbf{t}$, desired output length n, privacy parameter $\epsilon > 0$, rating function $\rho : V \times V \to [0, 1]$
Result: synthetic tf vector $\mathbf{s} \in \mathbb{N}^{|V|}$ with $|\mathbf{s}| = n$

```
for i ← 1 to n do            // produce output term-by-term
    v_i ←_R Cat(θ_t);                       // sample word v_i
    w_i ←_R E_{ε,ρ}(v_i);            // choose synonym for v_i
end
s ← (|{i ∈ [1, n] : w_i = w}|)_{w∈V};          // count synonyms
```

We keep the previous notation where V is the vocabulary of size L, $\mathbf{t} = (t_1, \dots, t_K)$ is the tf or tf–idf vector of the target document to be anonymized, and $\theta_\mathbf{t} := \mathbf{t}/\|\mathbf{t}\|_1$ is the corresponding vector of probabilities. For each pair of words $v, w \in V$, we have a similarity score $\rho(v, w) \in [0, 1]$. This score will be used in the Exponential mechanism, which outputs w on input v with probability

$$\pi_{v,w} := \Pr[\mathcal{E}_{\epsilon,\rho}(v) = w] = \frac{\exp\left(\frac{\epsilon}{2\Delta}\rho(v, w)\right)}{\sum_{w'} \exp\left(\frac{\epsilon}{2\Delta}\rho(v, w')\right)}.$$

Note that we assume that *all* potential inputs are adjacent which is a very conservative interpretation of differential privacy.

Our main result is that algorithm 1 is differentially private:

Theorem 3.2 (Differential Privacy of SynTF). *Given a privacy parameter $\epsilon > 0$ and an output length $n \in \mathbb{N}$, our SynTF mechanism (algorithm 1) fulfills ϵn-differential privacy.* □

The proof uses a counterpart of the known postprocessing lemma [12, proposition 2.1], which states that a convex combination of an ϵ-differentially private algorithm is again ϵ-differentially private.

3.4.1 Alternative Bound for the Exponential Mechanism. We can derive an alternative bound for the privacy loss of the Exponential mechanism by also considering the maximum change across *all outputs* for *fixed inputs* (in contrast to the sensitivity which tracks the maximum change across *adjacent inputs* for *fixed outputs*):

Theorem 3.3 (Alternative bound). *Let $\epsilon > 0$ be a privacy parameter and $\rho : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ be a rating function with sensitivity Δ and $|\mathcal{Z}| = L$. Let $\bar{\Delta} := \max_{x \in \mathcal{X}} \max_{z, z' \in \mathcal{Z}} |\rho(x, z) - \rho(x, z')|$. Then the privacy loss $\ell(\mathcal{E}_{\epsilon,\rho})$ is bounded by $(\bar{\epsilon} + \ln \eta)$, where*

$$\bar{\epsilon} := \epsilon \frac{\bar{\Delta}}{\Delta} \quad \text{and} \quad \eta = \eta(\bar{\epsilon}, L) = \frac{e^{-\bar{\epsilon}/2} + L - 1}{e^{\bar{\epsilon}/2} + L - 1} < 1. \quad \square$$

Typically, we will have $\bar{\Delta} > \Delta$ since the sensitivity Δ is restricted to *adjacent* inputs. The growth due to the factor $\bar{\Delta}/\Delta$ in $\bar{\epsilon} = \epsilon\bar{\Delta}/\Delta$ will therefore typically exceed the savings due to $\ln \eta < 0$, so the alternate bound $\bar{\epsilon} + \ln \eta$ will be *worse* than the original bound ϵ as derived in the standard differential privacy proof for the Exponential mechanism [27]. However, if we consider all inputs as adjacent, and if ρ is symmetric in its arguments, then we will have $\bar{\Delta} = \Delta$ and $\bar{\epsilon} = \epsilon$, and thus the factor $\eta < 1$ will provide a real improvement over the original bound. This is the case in our algorithm:

Corollary 3.4 (Improved differential privacy bound). *Given a privacy parameter $\epsilon > 0$ and an output length $n \in \mathbb{N}$, our SynTF mechanism fulfills $((\epsilon + \ln \eta(\epsilon, L)) \cdot n)$-differential privacy.* □

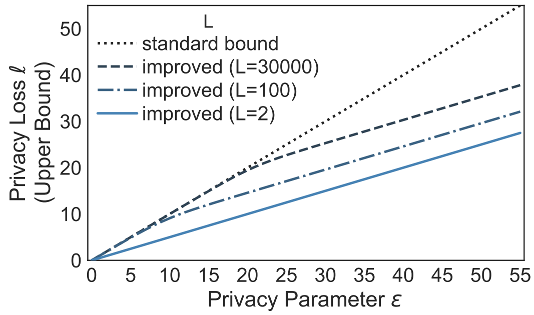


Figure 1: Standard and alternative upper bound $\epsilon + \ln \eta$ for the privacy loss $\ell(\mathcal{E}_{\epsilon,\rho})$ given different output space sizes L.

We illustrate the effects of the factor $\eta(\epsilon, L)$ in figure 1: The original upper bound ϵ is the black dotted line on top, the other lines show the improved upper bound $\epsilon + \ln \eta$ for different values of $L \in \{2, 100, 30000\}$. 30000 is approximately the size of the vocabulary in some of our experiments, The effect of the improved bound increases with the privacy parameter ϵ, whereas large output spaces have a smoothing effect that dampens the improvement.

3.4.2 Tight Worst-Case Bounds. A major factor in the differential privacy proof of theorem 3.3 and corollary 3.4 consists of bounding the privacy loss $\ell(\mathcal{E}_{\epsilon,\rho})$ for the Exponential mechanism used in algorithm 1. This privacy loss is defined as smallest upper bound for the fractions $\pi_{v_1,w}/\pi_{v_2,w}$, where $\pi_{v,w} \propto \exp\left(\frac{\epsilon}{2\Delta}\rho(v,w)\right)$ are the the associated probabilities. The probabilities $\pi_{v,w}$ depend on the underlying vocabulary V, the rating function ρ, and the privacy parameter ϵ, but do not take the documents $\mathbf{t}$ and $\mathbf{t}'$ into account. Therefore, we can compute the privacy loss

$$\ell(\mathcal{E}_{\epsilon,\rho}) = \max_{w \in V} \frac{\max_{v \in V} \pi_{v,w}}{\min_{v \in V} \pi_{v,w}}$$

in advance and independently from any documents to be anonymized once the parameters V, ρ, and ϵ have been determined. Our SynTF method with privacy parameter ϵ and output length n thus in fact fulfills ℓn- instead of ϵn-differential privacy where $\ell = \ell(\mathcal{E}_{\epsilon,\rho})$ is the privacy loss of the Exponential mechanism. This turns out to lead to huge gains in practice, reducing the privacy loss upper bound by almost 50% in our experiments (cf. section 4.2).

3.4.3 Necessary Condition on ϵ. The following theoretical result for the Exponential mechanism suggests that in order to get "useful" outputs with a large output space, we need to choose a large privacy parameter ϵ in the order of $\ln|\mathcal{Z}|$, under the assumption that there are only few good outputs for each input.

Corollary 3.5 (Necessary Condition on ϵ). *Let $\rho : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ be a rating function with sensitivity Δ and $|\mathcal{Z}| \in \mathbb{N}$. Take any fixed $x \in \mathcal{X}$ and denote by $\hat{\rho}_x$ and $\check{\rho}_x$ the maximum and minimum rating scores of $\rho(x, \cdot)$, respectively. For a desired minimum rating $\tau \in [\check{\rho}_x, \hat{\rho}_x]$, split $\mathcal{Z}$ into $\mathcal{T} := \{z \in \mathcal{Z} : \rho(x,z) \geq \tau\}$ and $\overline{\mathcal{T}} := \mathcal{Z} \setminus \mathcal{T}$. Given a probability $p \in [0,1]$, a necessary condition on ϵ for $\Pr[\mathcal{E}_{\epsilon,\rho}(x) \in \mathcal{T}] \geq p$ is*

$$\epsilon \geq \frac{2\Delta}{\hat{\rho}_x - \check{\rho}_x} \ln\left(\frac{p}{1-p} \cdot \frac{|\overline{\mathcal{T}}|}{|\mathcal{T}|}\right).$$

Note that for our SynTF algorithm, we have $\hat{\rho}_x - \check{\rho}_x \leq \Delta$. Hence for $p = 1/2$, the necessary condition becomes

$$\epsilon \geq 2\ln\left(\frac{p}{1-p} \cdot \frac{|\overline{\mathcal{T}}|}{|\mathcal{T}|}\right) = 2\ln\left(\frac{|\overline{\mathcal{T}}|}{|\mathcal{T}|}\right) = 2\ln\left(\frac{|\mathcal{Z}| - |\mathcal{T}|}{|\mathcal{T}|}\right).$$

Given a reasonable choice of τ, the number $|\mathcal{T}|$ of "useful" outputs whose score is at least τ will be small. In the case of our SynTF mechanism, we can think of τ as a threshold for the rating function that distinguishes good alternatives for a given word from poor ones, and $|\mathcal{T}|$ would reflect the number of suitable substitutes (synonyms). If we assume $|\mathcal{T}|$ to be bounded by some constant, then $\epsilon \in \Omega(\ln|\mathcal{Z}|)$, that is, ϵ *needs* to grow logarithmically in the size of the output space $|\mathcal{Z}|$ in order to allow meaningful results.

4 EVALUATION

In this section, we first describe our implementation of the SynTF mechanism along with associated parameters and our implementation choices. We then describe our experiment setup and report the evaluation results. Finally, we compare SynTF with a traditional information removal approach in the same experiment setup.

4.1 Algorithm Implementation and Parameters

We implemented a prototype of our SynTF algorithm in Python using the `SpaCy` package (http://spacy.io/) for text parsing functionality as well as the `numpy` and `SciPy` packages [17, 45] for (vector) computations. Besides the explicit parameters mentioned in algorithm 1, there are various implementation-dependent parameters that influence SynTF in its different stages. We now describe these parameters and corresponding implementation choices.

4.1.1 Vocabulary and Vectorization. We build a custom vectorizer to extract the vocabulary from the training or a given reference corpus, and to subsequently transform documents to their BoW tf vectors. We can specify several special options: Firstly, we can choose, for each extracted word, to keep its spelling as-is, to change its morphology through lemmatization, or to convert it to lower case. Secondly, we can instruct the vectorizer to include additional terms that are similar or synonymous to the actually extracted words, as to provide a greater choice of candidates for replacing a word with a suitable synonym but hopefully with different spelling to disturb lexical authorship attribution features. Our implementation uses the synonyms provided by WordNet's synsets. We remove stop words and numbers by default.

4.1.2 Similarity Rating Function. We now describe the rating function $\rho(v,w)$ that expresses the suitability of a substitute term w for an input term v. One fundamental technique are *word vectors* or *embeddings* which are dense vector representations of words in a real vector space. They are commonly derived with the intention that similar words have embeddings in the vector space that are nearby. We can therefore compute the similarity between two words simply and efficiently as *cosine similarity* between their corresponding word vectors. Two recent models to derive word vectors that achieve high accuracy in word similarity and analogy benchmarks are "word2vec" [28, 29] and "GloVe" [34].

As we saw in section 3.2, features such as the frequency of certain words and character n-grams often make an essential and decisive contribution to authorship attribution methods. Suppose we can choose a substitute for a given input term from a set of candidates with comparable similarity rating. Then to best prevent the attack, it is beneficial to pick the candidate that differs most in spelling from the input in order to obscure our word and n-gram frequencies. We can achieve this by including the (normalized) Levenshtein or n-gram distance in the rating function for the terms. Note that care must be taken to weight this appropriately – a too strong preference for differently-spelled substitutes will often pick completely different words that also have a different meaning from the original word, thus also negatively affecting the utility.

We have implemented the word similarity rating function as

$$\rho(v, w) := \cos(v, w) - sB(v, w),$$

where $\cos(v, w)$ is the cosine similarity between the corresponding GloVe [34] word vectors, and $B(v, w) \in [0, 1]$ is the *bigram overlap*, i.e. the proportion of matching letter bigrams in v and w. The scaling factor s determines if and how strong the bigram overlap affects the rating. As optimization, we precompute the word similarity ratings and probabilities for the Exponential mechanism for the entire vocabulary, which yields a significant performance boost.

4.2 Experiment Description

In this section, we describe the context and setup of our evaluation.

Dataset. We perform a series of experiments with our algorithm on the "20 newsgroups" dataset[1]. It comprises almost 19,000 postings from 20 different newsgroups, and comes with predefined train (60%) and test (40%) sets which we use throughout our experiments. For the text classification task, a label is provided for each message indicating the corresponding newsgroup. For the authorship attribution task, we extracted the "From" field in the header of each message and use it as author identifier. Note that we strip header and footer data before performing the actual classification and identification tasks as to make them more realistic.

Attack Scenarios. After filtering out missing and ambiguous identifiers, we count 5735 authors, but the majority provides insufficient training samples (below 20 for 5711 authors) for properly fitting a model. We therefore evaluate the attack only for the "top" authors with the largest number of messages in the dataset. Since the number of candidate suspects from which the correct author has to be determined also can influence the authorship attribution performance, we evaluate the attack for the *top 5* and *top 10* authors. Table 1 provides the number of train and test messages per author.

Another issue with the dataset is that some users are active in only a single newsgroup, in which case knowledge of authorship (attack) implies knowledge of the targeted newsgroup (utility). We therefore devise two subsets of authors:

Any Each suspect author can have postings in any number (one or more) of newsgroups.

Multi Each author must be active in *at least two* different newsgroups.

The idea of the "Multi" group is to reduce the similarity between the attacker's and analyst's tasks to allow a clearer distinction when evaluating the impact of our anonymization technique.

[1] http://qwone.com/~jason/20Newsgroups/

Table 1: Attack scenarios with minimum *per author* numbers for active groups and train/test messages in the dataset.

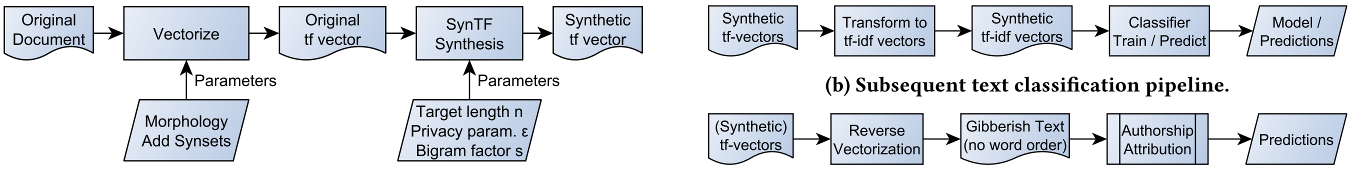
Scenario	Suspects	#Groups	#Train	#Test
Top 5/Any	Top 5	≥ 1	≥ 35	≥ 17
Top 10/Any	Top 10	≥ 1	≥ 28	≥ 9
Top 5/Multi	Top 5	≥ 2	≥ 29	≥ 9
Top 10/Multi	Top 10	≥ 2	≥ 21	≥ 8

Processing Pipeline. All documents traverse a processing pipeline that can be broken down into three parts: For each document, the main SynTF pipeline (figure 2a) first produces a synthetic tf vector (cf. section 3.3). It can be influenced by a number of parameters as described in section 4.1. Next, the synthetic tf vectors traverse the analyst's text classification pipeline (figure 2b) and the attacker's authorship attribution pipeline (figure 2c) to measure the prediction performance for each task. In both cases, we evaluate a multinomial naïve Bayes classifier and a linear SVM. We perform 10 runs of the entire pipeline (anonymization + evaluation) for each combination of parameters to reduce fluctuations and get stable results.

The analyst (cf. figure 2b) first transforms the tf vectors to tf–idf vectors which are commonly used in classification tasks. He then trains a classifier with the *training* subset of the dataset, and subsequently uses it to predict the newsgroups for the *test* subset. We implement the classification in Python based on `scikit-learn` [33], using its `MultinomialNB` classifier with smoothing ($\alpha = 0.01$), and its `LinearSVC` classifier with default parameters ($C = 1$).

For the attack, we make use of the "JStylo" authorship attribution framework [26]. It supports several extended feature sets such as "WritePrints" proposed in [1]. WritePrints includes additional stylistic features (cf. section 3.2) on top of the usual BoW that have to be extracted from full texts. However, since the attacker only gets synthetic tf vectors and not full texts, she first converts the numbers in the tf vectors to text by repeating each word accordingly, which allows at least partial deduction of WritePrints features ("reverse vectorization" in figure 2c). Note that the "full" WritePrints feature set contains a virtually endless number of features and severely degrades performance (speed). Furthermore, the authors of [26] have shown that despite its title, the "limited" version even outperforms the "full" WritePrints in terms of accuracy, which we could confirm in own experiments. Therefore, we keep the default JStylo configuration with the "WritePrints (Limited)" feature set. JStylo builds on the Weka machine learning library. We use its `NaiveBayesMultinomial` classifier with Laplace smoothing and its `SMO` SVM classifier with linear kernel and $C = 1$ by default.

Finding Optimal Parameters. We perform a grid search over the SynTF parameters listed in table 2 to find "optimal" parameters in the sense that they should simultaneously strongly affect authorship attribution but mostly leave classification into newsgroups unaffected. As metric to find these optimal settings we use the difference between the relative performance impacts on utility and attack: Given parameters $\mathbf{p}$, denote by $\beta_U(\mathbf{p})$ the relative performance of the analyst's classification task (measured as F_1 score), and similarly denote by $\beta_A(\mathbf{p})$ the relative performance of the attacker's task. Then the optimal parameters are $\hat{\mathbf{p}} = \operatorname{argmax}_{\mathbf{p}}(\beta_U(\mathbf{p}) - \beta_A(\mathbf{p}))$.



(a) The main SynTF processing pipeline, including parameters.

(b) Subsequent text classification pipeline.

(c) Subsequent authorship attribution pipeline.

Figure 2: Processing pipelines for the main SynTF mechanism and subsequent analyst and attacker tasks.

Table 2: Evaluated and optimal SynTF parameters.

Parameter	Values	Description
morpho-logy	lemma	Lemmatize words.
	lower	Convert words to lower case.
	orth	Leave spelling unchanged.
synsets	true/false	Extend vocabulary with additional synonyms from WordNet.
s	0, 0.1, 0.2, 0.3, 0.4	Impact factor of letter bigram overlap on rating function ρ.
n	100, 150, 200	Length of output vector (words).
ϵ	35–55 (47.5), effectively 25.4	Privacy parameter (stepsize 2.5). Effective loss ℓ, cf. sec. 3.4.2.

Since we want them to equally cover all four attack scenarios, we find optimal parameters that maximize the *minimum* difference $\beta_U(\mathbf{p}) - \beta_A(\mathbf{p})$ over all attack scenarios. Furthermore, we perform 10 runs of the anonymization–evaluation process for each combination of parameters to reduce fluctuations and get stable results.

4.3 Discussion of Results

After running the evaluation, we found the optimal parameters highlighted in table 2 with privacy parameter $\epsilon = 47.5$. However, our tight bounds analysis (cf. section 3.4.2) shows that the effective privacy loss $\ell(\mathcal{E}_{\epsilon,\rho}) \approx 25.4$ is only about half as large. Table 3 provides exemplary performance figures in the “Top 10/Any” scenario for both topic classification and authorship attribution. Figure 3 depicts the relative performance between utility (green lines, left y-axis) and attack (red lines, right y-axis) in the different stages of SynTF. The bottom x-axis indicates the privacy parameter ϵ, with the corresponding effective privacy loss values $\ell(\mathcal{E}_{\epsilon,\rho})$ on the top. The dotted, dashed, and solid lines mark the utility and attack performances with the original (plaintext), vectorized, and synthetic data, respectively, where we used the optimal parameters for vectorization and synthesis as mentioned above.

We observe that the vectorization already affects the attack more due to the loss of structural and syntactic features, except in one case (Top 5/Multi). Note that the size of the (positive) gap between the green and red lines indicate the analyst’s gain over the attacker in terms of the relative performance of the corresponding stage of the anonymization. Obviously both utility and attack suffer with a decreasing privacy parameter ϵ. However, in most cases the gap between analyst and attacker is even higher than after vectorization, which indicates a growing advantage for the analyst. Furthermore, it shows that our SynTF mechanism successfully impairs authorship attribution while having only a mild effect on the classification task.

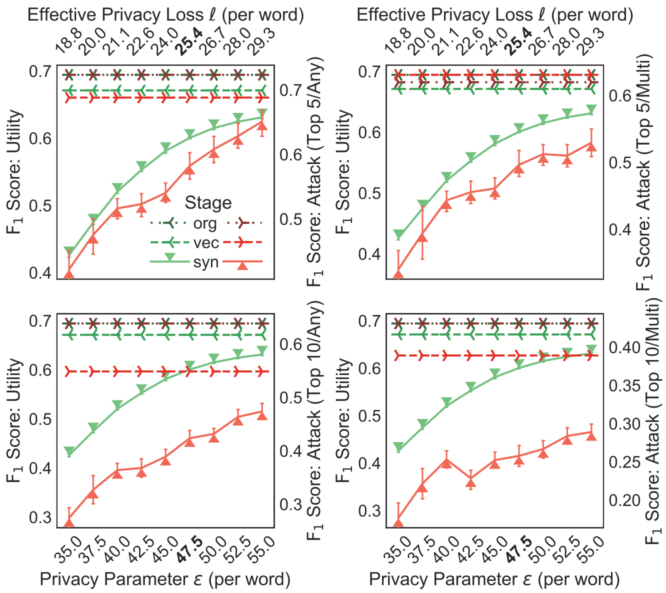
Impact of Attack Scenarios. Comparing the four scenarios with respect to the gap size, we make the following deductions: As expected, authorship attribution quickly becomes harder with an increasing number of suspect authors. Similarly, excluding authors who are active in only one newsgroup widens the gap, as we can see when going from the “Any” to the “Multi” scenarios. This indicates that our method is even more effective when the benign and malicious tasks are actually based on *distinct* problems.

Impact of Parameters from Table 2. A key factor in the success of our method is the letter bigram overlap B in the rating function ρ. Its effect of preferring synonyms with different spelling improves the capability of our method to prevent authorship attribution attacks. We illustrate this effect depending on the bigram overlap factor s in figure 4: Without bigram overlap ($s = 0$), the attacker has an advantage in all “Top 5” scenarios (red bars). Only when $s \geq 0.3$, we see a shift of power in favor of the analyst (green bars). In the “Top 10” scenarios, the analyst enjoys an advantage even without the bigram overlap, but we can roughly double his advantage if we choose the optimal value $s = 0.3$.

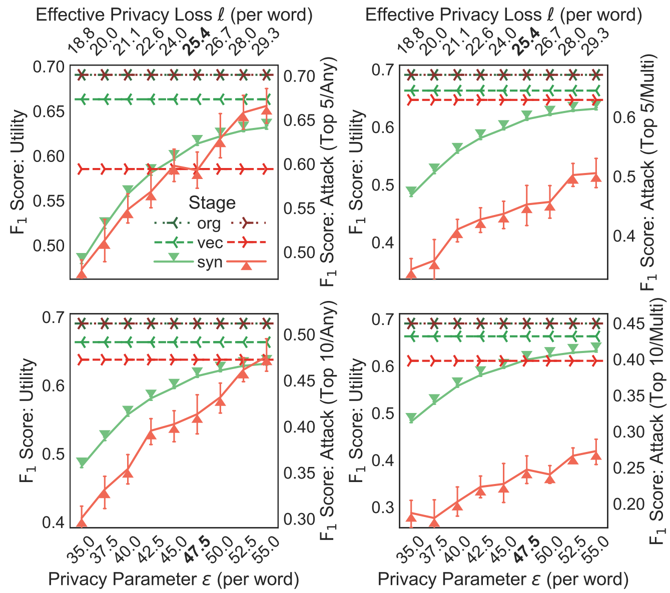
Regarding morphology, observe that the use of upper and lower case letters is a stylistic feature that can pose a clue for authorship attribution but barely has any relevance for topic inference. Therefore, transforming all words to lowercase affects the attacker more than the analyst. Lemmatization strips off word endings and hence reduces the attacker’s information on writing style further, but it also has an impact on classification since the meaning can change between a word and its lemma. Still, in terms of our definition of “optimal” parameters, using lemmatized words gave the best relative performance gain for the analyst, indicating that the lost word endings are more severe for the attack.

Other parameters are less insightful: Increasing the output length will help increase both tasks’ performance, however, the gain becomes less for larger output lengths. Moreover, the inclusion of additional synonyms in the vocabulary did not provide any benefit.

SVM Anomaly. We observe one anomaly in the “Top 5/Any” scenario for the SVM. Apparently, vectorization already causes a drastic reduction of the attack performance. However, for $\epsilon \geq 45$, going from vectorized to synthetic vectors *increases* the attack performance. This is unexpected since the information *lost in vectorization* will not be restored by the synthesis process. Our current hypothesis is that the SVM might overfit on the vectorized training data, causing poor predictions on the vectorized test data, and the randomness in the synthesis step in turn acts as regularization.



(a) Multinomial naïve Bayes.



(b) Linear SVM.

Figure 3: Relative performance of analyst (green) and attacker (red) tasks in different stages of the SynTF process, per attack scenario (dotted: original data, dashed: tf vectors, solid: synthetic tf vectors). A (positive) gap between the green and red lines shows how much the attack is more affected than utility. Impact on attack increases with number of authors and active groups.

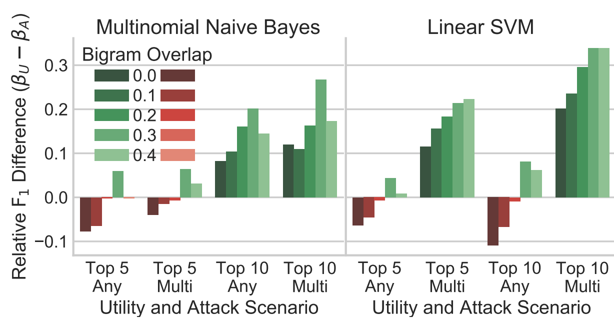


Figure 4: Impact of letter bigram overlap factor s.

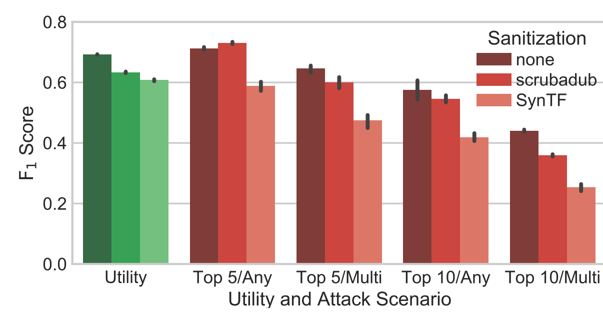


Figure 5: Comparing SynTF and traditional data removal.

Table 3: Evaluation results (Top 10/Any)

	Utility			Attack			Gain
Method	F_1	P	R	F_1	P	R	ΔF_1
none (original)	0.69	0.71	0.70	0.64	0.71	0.63	0.06
SynTF abs.	0.60	0.61	0.61	0.42	0.44	0.43	0.18
scrubadub abs.	0.64	0.65	0.65	0.57	0.63	0.57	0.06
SynTF rel.	87%	86%	87%	66%	61%	69%	20%
scrubadub rel.	92%	92%	92%	90%	88%	91%	02%

4.3.1 Comparison with Scrubbing Methods. We run the open source scrubadub (http://scrubadub.readthedocs.org/) tool on the 20 newsgroups dataset to remove PII and evaluate the utility and attack performance in our scenarios. Figure 5 shows a comparison of the results with our SynTF method and optimal parameters. The results indicate that our method outperforms the scrubbing technique in preventing the attack in all four attack scenarios, at a comparable level of utility. For instance, in the "Top 10/Any" scenario listed in table 3, SynTF achieves an F_1 score of 0.60 for classification, where scrubadub is slightly better with 0.64, down from 0.69. For the attack, however, scrubadub drops from 0.64 to 0.57, whereas SynTF manages to more than triple the reduction and push the attacker's performance down to 0.42.

5 RELATED WORK

Authorship Obfuscation. Several countermeasures against authorship attribution have been proposed. Rao and Rohatgi [35] examine newsgroups postings and identify the authors from the body of the text by analyzing the frequency of function words. They suggest to either use automated *machine translation* to a foreign language and back, or to *educate authors* who want to write anonymous documents about authorship attribution attacks. However, these countermeasures are insufficient: In [7], Caliskan et al. show that authorship attribution is still possible even after performing multiple machine translations. Furthermore, Afroz et al. [3] show that *deceptive writing* by an author trying to imitate another or to obfuscate his own writing style can be detected with high accuracy.

Anonymouth [26] is based on JStylo and uses clustering of two references sets with the author's and foreign sample texts to propose manual changes that have to be made to the document. The process must be repeated until authorship attribution is prevented sufficiently. Kacmarcik and Gamon [19] follow a similar but more automated approach based on decision trees and SVMs. Their method adjusts the tf vector of a document by moving its feature values closer to those of other writers, as to prevent the classifier from identifying the correct author. While the countermeasure is effective against the evaluated SVMs with up to 70 features, the more sophisticated *unmasking* approach by Koppel and Schler [21, 22] is still able to distinguish the actual author from others. Kacmarcik and Gamon in turn propose a "deep obfuscation" variant of their method which iteratively tries to make unmasking harder, however, this requires more and more changes to be made to the documents.

The results indicate that both methods are successful in preventing authorship attribution attacks in theory. However, the authors of Anonymouth [26] observed that while users were able to implement the suggested changes for very small feature sets with only 9 features, they were overstrained with the changes for the more complex "WritePrints (Limited)" feature set which we also used in our experiments. Similarly, Kacmarcik and Gamon [19] observed that for a deep level of obfuscation, one would have to consider more and more features and make corresponding changes to the document, thus increasing the complexity for the user. In practice, both methods seem cumbersome for the user if a deep level of obfuscation shall be reached. Furthermore, both methods only prevent authorship attribution with respect to a *specific* reference corpus with other authors. While our method does not produce human-readable texts, it requires no manual changes to the documents, and its protection is independent of a reference corpus.

De-Identification. De-identification (or *scrubbing*) methods provide a way to remove personally identifiable information (PII) from textual documents. They are often motivated by the health care and medical sectors and focus on identifying and removing particular types of personal information such as protected health information (PHI), a list of 18 identifiers as specified in the US Health Insurance Portability and Accountability Act (HIPAA) [43]. Popular methods include the "Scrub System" [41], the "MITRE Identification Scrubber Toolkit" (MIST) [2], or the PhysioNet "deid" software package [32]. They typically work with lists of names and identifiers, regular expressions, simple heuristics, and also machine learning techniques to identify and remove pieces of text that constitute PII.

While this kind of information must be removed to protect the privacy of the subjects mentioned in the document, our experiments in section 4.3.1 show that de-identification based on scrubbing provides no adequate protection for the document's author although this is often critical, as in the case of complaint letters or patient records to protect the privacy of the treating physician. Moreover, we found that publications on these methods typically only evaluate their methods' ability to identify and remove all pieces of PII in the text (cf. the survey by Uzuner et al. [44]). We have not seen any evaluation on the impact of scrubbing on further processing with text mining techniques such as document classification, and more importantly, we have not found an evaluation whether and to what extend these methods prevent authorship attribution techniques.

Differential Privacy. Differential privacy has been successfully applied to a wide range of problems from simple statistical functions to machine learning. The survey by Dwork [10] provides a good overview of some earlier results. It is commonly used to provide *aggregate* statistics, that is, multiple records are combined into one result. A good example is RAPPOR [13], which allows the collection of anonymized user statistics even over time. However, releasing aggregate information only allows inferences on an entire population, whereas we want to classify each document individually. Releasing *individual* data with an ϵ comparable to aggregating mechanisms causes too much noise for individual records as it masks any difference (topic, sentiment, etc.) between two inputs and hence prevents any utility. The issue is well-known in the literature and has been observed e.g. in the context of locations [4, 25], graphs [37], and recommender systems [24]. Approaches typically involve relaxing the privacy- or adjacency-definition [4, 8, 15]. Andrés et al. [4] circumvent the issue for location data by generalizing differential privacy to metrics [8]. For graphs, Hay et al. [14] define two varians of differential privacy, namely *node* and *edge* privacy, where two graphs are considered adjacent if they differ either in an entire node (including its edges) or in just a single edge. According to Kasiviswanathan et al. [20], most works focus on the strictly weaker edge privacy since it is harder to create node private algorithms providing good utility with a comparable privacy loss. For instance, Sala et al. [37] revert to edge privacy for sharing graphs and obtain usable results with $\epsilon = 100$ *per edge* (instead of *per node*). In comparison, our SynTF mechanism achieves a privacy loss of only 25.4 *per word* in the output (instead of *per document*).

6 CONCLUSION

We have presented SynTF, a novel approach to produce anonymized, synthetic term frequency vectors which can be used in lieu of the original term frequency vectors in typical applications based on the vector space model. Our method produces sparse vectors which are favorable regarding performance and memory efficiency. We have proved that our method fulfills differential privacy which currently serves as a "gold standard" for privacy definitions. Since our method anonymizes each text individually, it can be used locally at the data source to anonymize documents on-premise before collection, e.g., to obtain anonymized training data for machine learning or provide personalized ads based on anonymized emails or search queries.

Although our method requires a large ϵ to get reasonable utility, we provide evidence that this is necessary: First, we want to be able to analyze records independently from each other, thus the anonymization must *not* hide the influence of individual records in the result. Second, we have derived a necessary condition on the privacy parameter ϵ for the Exponential mechanism indicating that it must grow logarithmically in the size of the output space when high utility is required but only a limited number of "good" outputs is available. To further address the issue, we have derived alternative bounds on the privacy loss of the Exponential mechanism, which in our case provide a substantial reduction of almost 50%.

We have performed an extensive evaluation of SynTF on the 20 newsgroups dataset and analyzed the influence of different parameters. Our results indicate that it effectively prevents authorship attribution while facilitating tasks such as classification (utility). In contrast, our experiments show that traditional scrubbing methods are insufficient at preventing authorship attribution attacks.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 653497 (PANORAMIX).

REFERENCES

[1] A. Abbasi and H. Chen. 2008. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems (TOIS)* 26, 2 (2008), 7.

[2] J. Aberdeen, S. Bayer, R. Yeniterzi, B. Wellner, C. Clark, D. Hanauer, B. Malin, and L. Hirschman. 2010. The MITRE Identification Scrubber Toolkit: design, training, and assessment. *International journal of medical informatics* 79, 12 (2010), 849–859.

[3] S. Afroz, M. Brennan, and R. Greenstadt. 2012. Detecting hoaxes, frauds, and deception in writing style online. In *2012 IEEE Symposium on Security and Privacy.* IEEE, 461–475.

[4] M. Andrés, N. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM, 901–914.

[5] M. Barbaro, T. Zeller, and S. Hansell. 2006. A face is exposed for AOL searcher no. 4417749. *New York Times* 9, 2008 (9 August 2006), 8For.

[6] S. Busemann, S. Schmeier, and R.G. Arens. 2000. Message classification in the call center. In *Proceedings of the sixth conference on Applied natural language processing.* Association for Computational Linguistics, 158–165.

[7] A. Caliskan and R. Greenstadt. 2012. Translate once, translate twice, translate thrice and attribute: Identifying authors and machine translation tools in translated text. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on.* IEEE, 121–125.

[8] K. Chatzikokolakis, M. Andrés, N. Bordenabe, and C. Palamidessi. 2013. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium.* Springer, 82–102.

[9] Y.-A. De Montjoye, C.A. Hidalgo, M. Verleysen, and V.D. Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3 (2013).

[10] C. Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation.* Springer, 1–19.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference.* Springer, 265–284.

[12] C. Dwork, A. Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[13] Ú. Erlingsson, V. Pihur, and A. Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security.* ACM, 1054–1067.

[14] M. Hay, C. Li, G. Miklau, and D. Jensen. 2009. Accurate estimation of the degree distribution of private networks. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on.* IEEE, 169–178.

[15] X. He, A. Machanavajjhala, and B. Ding. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data.* ACM, 1447–1458.

[16] M. Jawurek, M. Johns, and K. Rieck. 2011. Smart metering de-pseudonymization. In *Proceedings of the 27th Annual Computer Security Applications Conference.* ACM, 227–236.

[17] E. Jones, T. Oliphant, P. Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[18] P. Juola, J. Sofko, and P. Brennan. 2006. A prototype for authorship attribution studies. *Literary and Linguistic Computing* 21, 2 (2006), 169–178.

[19] G. Kacmarcik and M. Gamon. 2006. Obfuscating document stylometry to preserve author anonymity. In *Proceedings of the COLING/ACL on Main conference poster sessions.* Association for Computational Linguistics, 444–451.

[20] S.P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography.* Springer, 457–476.

[21] M. Koppel and J. Schler. 2004. Authorship verification as a one-class classification problem. In *Proceedings of the twenty-first international conference on Machine learning.* ACM, 62.

[22] M. Koppel, J. Schler, and E. Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research* 8, Jun (2007), 1261–1276.

[23] B. Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

[24] A. Machanavajjhala, A. Korolova, and A.D. Sarma. 2011. Personalized social recommendations: accurate or private. *Proceedings of the VLDB Endowment* 4, 7 (2011), 440–450.

[25] D. Machanavajjhala, A.and Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. 2008. Privacy: Theory meets practice on the map. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering.* IEEE Computer Society, 277–286.

[26] A.W. McDonald, S. Afroz, A. Caliskan, A. Stolerman, and R. Greenstadt. 2012. Use fewer instances of the letter "i": Toward writing style anonymization. In *International Symposium on Privacy Enhancing Technologies Symposium.* Springer, 299–318.

[27] F. McSherry and K. Talwar. 2007. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on.* IEEE, 94–103.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[29] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.

[30] F. Mosteller and D.L. Wallace. 1963. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *J. Amer. Statist. Assoc.* 58, 302 (1963), 275–309.

[31] A. Narayanan and V. Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008).* IEEE, 111–125.

[32] I. Neamatullah, M.M. Douglass, H.L. Li-wei, A. Reisner, M. Villarroel, W.J. Long, P. Szolovits, G.B. Moody, R.G. Mark, and G.D. Clifford. 2008. Automated de-identification of free-text medical records. *BMC medical informatics and decision making* 8, 1 (2008), 32.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[34] J. Pennington, R. Socher, and C.D. Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–43.

[35] J.R. Rao, P. Rohatgi, et al. 2000. Can pseudonymity really guarantee privacy?. In *USENIX Security Symposium.* 85–96.

[36] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. 1998. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, Vol. 62. 98–105.

[37] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B.Y. Zhao. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference.* ACM, 81–98.

[38] G. Salton, A. Wong, and C.S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[39] H. Schütze, C.D. Manning, and P. Raghavan. 2008. *Introduction to Information Retrieval.* Vol. 39. Cambridge University Press.

[40] E. Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology* 60, 3 (2009), 538–556.

[41] L. Sweeney. 1996. Replacing personally-identifying information in medical records, the Scrub system.. In *Proceedings of the AMIA annual fall symposium.* American Medical Informatics Association, 333.

[42] L. Sweeney. 2000. Simple demographics often identify people uniquely. *Health (San Francisco)* 671 (2000), 1–34.

[43] U.S. Dept. of Labor, Employee Benefits Security Administration. 1996. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). , 191 pages. http://www.hhs.gov/hipaa/

[44] Ö. Uzuner, Y. Luo, and P. Szolovits. 2007. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association* 14, 5 (2007), 550–563.

[45] S.v.d. Walt, S.C. Colbert, and G. Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22–30.

[46] B. Weggenmann and F. Kerschbaum. 2018. SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. *arXiv preprint arXiv:1805.00904* (2018). http://arxiv.org/abs/1805.00904

[47] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett. 2013. Differentially private histogram publication. *The VLDB Journal* 22, 6 (2013), 797–822.