A Girl Has A Name: Detecting Authorship Obfuscation

Asad Mahmood **Zubair Shafiq** **Padmini Srinivasan**

The University of Iowa

{asad-mahmood,zubair-shafiq,padmini-srinivasan}@uiowa.edu

Abstract

Authorship attribution aims to identify the author of a text based on the stylometric analysis. *Authorship obfuscation*, on the other hand, aims to protect against authorship attribution by modifying a text's style. In this paper, we evaluate the stealthiness of state-of-the-art authorship obfuscation methods under an adversarial threat model. An obfuscator is *stealthy* to the extent an adversary finds it challenging to detect whether or not a text modified by the obfuscator is obfuscated – a decision that is key to the adversary interested in authorship attribution. We show that the existing authorship obfuscation methods are not stealthy as their obfuscated texts can be identified with an average F1 score of 0.87. The reason for the lack of stealthiness is that these obfuscators degrade text smoothness, as ascertained by neural language models, in a detectable manner. Our results highlight the need to develop stealthy authorship obfuscation methods that can better protect the identity of an author seeking anonymity.

1 Introduction

Authorship attribution aims to identify the author of a text using stylometric techniques designed to capitalize on differences in the writing style of different authors. Owing to recent advances in machine learning, authorship attribution methods can now identify authors with impressive accuracy (Abbasi and Chen, 2008) even in challenging settings such as cross-domain (Overdorf and Greenstadt, 2016) and at a large-scale (Narayanan et al., 2012; Ruder et al., 2016). Such powerful authorship attribution methods pose a threat to privacy-conscious users such as journalists and activists who may wish to publish anonymously (Times, 2018; Anonymous, 2018).

Authorship obfuscation, a protective countermeasure, aims to evade authorship attribution by obfuscating the writing style in a text. Since it is challenging to accomplish this manually, researchers have developed automated authorship obfuscation methods that can evade attribution while preserving semantics (PAN, 2018). However, a key limitation of prior work is that authorship obfuscation methods do not consider the adversarial threat model where the adversary is "obfuscation aware" (Karadzhov et al., 2017; Potthast et al., 2018; Mahmood et al., 2019). Thus, in addition to evading attribution and preserving semantics, it is important that authorship obfuscation methods are "stealthy" – i.e., they need to hide the fact that text was obfuscated from the adversary.

In this paper, we investigate the stealthiness of state-of-the-art authorship obfuscation methods. Our intuition is that the application of authorship obfuscation results in subtle differences in text smoothness (as compared to human writing) that can be exploited for obfuscation detection. To capitalize on this intuition, we use off-the-shelf pre-trained neural language models such as BERT and GPT-2 to extract text smoothness features in terms of word likelihood. We then use these as features to train supervised machine learning classifiers. The results show that we can accurately detect whether or not a text is obfuscated.

Our findings highlight that existing authorship obfuscation methods themselves leave behind stylistic signatures that can be detected using neural language models. Our results motivate future research on developing stealthy authorship obfuscation methods for the adversarial threat model where the adversary is obfuscation aware.

Our key contributions are as follows:

- We study the problem of obfuscation detection for state-of-the-art authorship obfuscation methods. This and the underlying property of stealthiness has been given scant attention in the literature. We also note that this problem is potentially more challenging

than the related one of synthetic text detection since most of the original text can be retained during obfuscation.

- We explore 160 distinct BERT and GPT-2 based neural language model architectures designed to leverage text smoothness for obfuscation detection.

- We conduct a comprehensive evaluation of these architectures on 2 different datasets. Our best architecture achieves F1 of 0.87, on average, demonstrating the serious lack of stealthiness of existing authorship obfuscation methods.

Paper Organization: The rest of this paper proceeds as follows. Section 2 summarizes related work on authorship obfuscation and obfuscation detection. Section 3 presents our proposed approach for obfuscation detection using neural language models. Section 4 presents details of our experimental setup including the description of various authorship obfuscation and obfuscation detection methods. We present the experimental results in Section 5 before concluding. The relevant source code and data are available at https://github.com/asad1996172/Obfuscation-Detection.

2 Related Work

In this section, we separately discuss prior work on authorship obfuscation and obfuscation detection.

2.1 Authorship Obfuscation

Given the privacy threat posed by powerful authorship attribution methods, researchers have started to explore text obfuscation as a countermeasure. Early work by Brennan et al. (2012) instructed users to manually obfuscate text such as by imitating the writing style of someone else. Anonymouth (McDonald et al., 2012, 2013) was proposed to automatically identify the words and phrases that were most revealing of an author's identity so that these could be manually obfuscated by users. Follow up research leveraged automated machine translation to suggest alternative sentences that can be further tweaked by users (Almishari et al., 2014; Keswani et al., 2016). Unfortunately, these methods are not effective or scalable because it is challenging to manually obfuscate text even with some guidance.

Moving towards full automation, the digital text forensics community (Potthast and Hagen, 2018) has developed rule-based authorship obfuscators (Mansoorizadeh et al., 2016; Karadzhov et al., 2017; Castro-Castro et al., 2017). For example, Karadzhov et al. (2017) presented a rule-based obfuscation approach to adapt the style of a text towards the "average style" of the text corpus. Castro et al. (2017) presented another rule-based obfuscation approach to "simplify" the style of a text.

Researchers have also proposed search and model based approaches for authorship obfuscation. For example, Mahmood et al. (2019) proposed a genetic algorithm approach to "search" for words that when changed, using a sentiment-preserving word embedding, would have the maximum adverse effect on authorship attribution. Bevendorff et al. (2019) proposed a heuristic-based search algorithm to find words that when changed using operators such as synonyms or hypernyms, increased the stylistic distance to the author's text corpus. Shetty et al. (2018) used Generative Adversarial Networks (GANs) to "transfer" the style of an input text to a target style. Emmery et al. (2018) used auto-encoders with a gradient reversal layer to "de-style" an input text (aka style invariance).

2.2 Obfuscation Detection

Prior work has successfully used stylometric analysis to detect *manual* authorship obfuscation (Juola, 2012; Afroz et al., 2012). The intuition is that humans tend to follow a particular style as they try to obfuscate a text. In a related area, Shahid et al. (2017) used stylometric analysis to detect whether or not a document was "spun" by text spinners. We show later that these stylometric-methods do not accurately detect more advanced automated authorship obfuscation methods.

There is increasing interest in distinguishing synthetic text generated using deep learning based language models such as BERT and GPT-2 from human written text. Using contextual word likelihoods, as estimated using a pre-trained language model (Radford et al., 2019), Gehrmann et al. (2019) were able to raise the accuracy of humans at detecting synthetic text from 54% to 72%. Zellers et al. (2019) showed that a classifier based on a language model can accurately detect synthetic text generated by the same language model.

However, the detection accuracy degrades when different language models are used to generate and to detect. Bakhtin et al. (2019) also showed that the detection accuracy degrades when the synthetic text is generated using a language model trained on a different corpus.

In summary, recent research has leveraged language models to detect their generated synthetic text. However, in obfuscation we start with human written text and make modifications such that text semantics is still preserved. This is in part achieved by retaining chunks of the original writing. Thus, the quirks of the obfuscator will be mingled in unpredictable proportions and ways with the author's original writing style. This makes the detection of obfuscated text different and potentially more challenging than synthetic text detection. To the best of our knowledge, this work presents the first systematic study of the detection of automatically obfuscated text.

3 Proposed Approach

3.1 Intuition

An automated authorship obfuscator changes the input text so that it evades authorship attribution while preserving semantics. The quality and smoothness of automated text transformations using the state-of-the-art obfuscators differ from that of human written text (Mahmood et al., 2019). Therefore, the intuition behind our obfuscation detectors is to exploit the differences in text smoothness between human written and obfuscated texts. We capture text smoothness using powerful pre-trained context aware neural language models.[1] A text with a relatively greater proportion of high likelihood words is likely to be more smooth.

3.2 Detector Architectures

Figure 1 shows the pipeline of our method for detecting whether or not a given text is obfuscated. First, a language model is used to extract the likelihood (in the form of probability or rank) for each word in the text. Second, these likelihoods are used to build a smoothness representation for the text. This is input to a supervised machine learning model that is trained to classify the text as human written or obfuscated. The three steps correspond to three significant architectural dimensions of our detectors with multiple algorithmic options in each dimension. Combinations of choices along each dimension yield different architectures that can be used by an adversary to detect obfuscated documents. We detail each dimension next.

3.2.1 Word likelihood extraction

Given a word sequence, language models are designed to predict the next word. They do this by building contextual models of word occurrences as probability distributions over the full vocabulary. Then some heuristic is used to pick the next word e.g., select the word with the highest probability. In our case, instead of word prediction, we extract the likelihood from the language model (either as a probability or as a rank) for each word in the text given its context.

The language model has a critical role. Thus, we use neural language models with deep architectures and trained on large amounts of data which are better at identifying both long-term and short-term context. In order to imitate an adversary who may not have the significant resources needed to train such models, we use off-the-shelf pre-trained neural language models. Specifically, we choose well-known context-aware neural language models GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2018). We choose both as they use different approaches. GPT-2 has been shown to perform better than BERT (Gehrmann et al., 2019) at synthetic text detection, with word rank giving higher performance than word probability. Their relative merit for obfuscation detection is unknown.

1) GPT-2. GPT-2 released by Open AI in 2019 uses at its core, a variation of the "transformer" architecture, an attention based model (Vaswani et al., 2017) and is trained on text from 45 million outbound links on Reddit (40 GB worth of text). We use GPT-2 to compute the conditional probability for word i as $p(w_i|w_{1...i-1})$. The position of w_i in the sorted list (descending order of probability) of vocabulary words gives the word rank. The authors (Radford et al., 2019) trained four versions of GPT-2 differing in architecture size. Of these, we used the small and medium versions containing 117M and 345M parameters, respectively. The authors eventually also released a large version containing 762M parameters and a very large version containing 1542M parameters.[2] We did not use

[1] BERT: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html;
GPT-2: https://openai.com/blog/better-language-models

[2] https://openai.com/blog/gpt-2-6-month-follow-up/

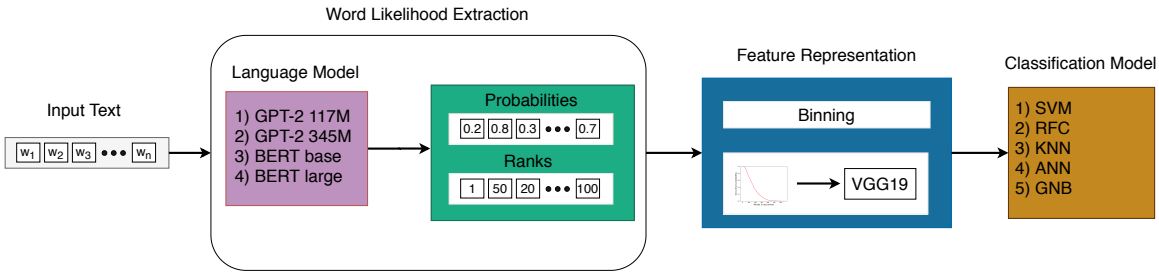

Figure 1: Pipeline for obfuscation detection

them because only the small and medium versions were released at the time of our experimentation.

2) BERT. BERT released by Google in 2018 is also based on "Transformers". It is trained on text from Wikipedia (2.5B words) and BookCorpus (800M words). BERT considers a bi-directional context unlike the uni-directional context considered by GPT-2. Thus, in BERT the conditional occurrence probability for word i is $p(w_i|w_{i-k...i-1}, w_{i+1...i+k})$ where k is the window size on each direction. Rank is computed in the similar way as GPT-2. We use both pre-trained BERT: BERT BASE with 110M parameters and BERT LARGE with 340M parameters.

We implement likelihood extraction for both GPT-2 and BERT, using code made available by the Giant Language Model Test Room (GLTR) tool.[3]

3.2.2 Feature Representation

We experiment with two different representations of smoothness. Each is explored with occurrence probabilities and with ranks.

1) Binning based features: Text smoothness is represented by the likelihood of words in text. A text with a greater proportion of high likelihood words is likely to be smoother. We aggregate this information using fixed size bins representing different likelihood ranges. For probabilities we create bin sizes of 0.001, 0.005 and 0.010. For ranks we create bin sizes of 10, 50 and 100. Thus for example, one feature representation is to consider bins of ranks from 0 to 10, 11 to 20, 21 to 30 etc. Each bin contains the proportion of words in the document with likelihood in that range.

2) Image based features: Since the word likelihood values received from language models are in essence signals, we explore signal detection approaches as well. For example, for audio classification (Hershey et al., 2017) store plots of the log-mel spectogram of the audios as images and then apply image classification methods. VGG (Simonyan and Zisserman, 2014), was one of the top performers of the different classifiers they tested. Inspired by them, we explore obfuscation detection via image classification. Specifically, we explore a transfer learning approach wherein we use the VGG-19 classifier[4] trained for image classification on ImageNet dataset[5]. For our method, we sort the extracted likelihood values for the text in descending order and then plot these values saving it as an image. This image is then processed by the pre-trained VGG-19. We extract the document's [6] representation from the last *flatten* layer of VGG-19 (before the fully connected layers) as it contains high-level information regarding edges and patterns in the image. We expect this resulting feature representation vector to capture information regarding text smoothness.

3.2.3 Classification

We experiment with Support Vector Machine (SVM) with a linear kernel, Random Forest Classifier (RFC) an ensemble learning method, K Nearest Neighbor (KNN) which is a non-parametric method, Artificial Neural Network (ANN) which is a parametric method, and Gaussian Naive Bayes (GNB) which is a probabilistic method. All classifiers are trained using default parameters from scikit-learn[7] except for ANN, where we use *lbfgs* solver instead of *adam* because it is more performant and works well on smaller datasets.

3.2.4 Detection Architectures

Options selected for each dimension combine to form a distinct obfuscation detection architecture.

[3]https://github.com/HendrikStrobelt/detecting-fake-text

[4]https://keras.io/applications/#vgg19

[5]http://www.image-net.org/

[6]Terms 'text' and 'document' are used interchangeably

[7]https://scikit-learn.org/stable/

With 4 language models giving probabilities or ranks as output, 4 features (3 binning based features and 1 image based feature) and 5 different classifiers we experiment with a total of 160 distinct architectures. The assumption here is that a determined adversary will similarly look for the most effective obfuscation detector.

4 Experimental Setup

4.1 Authorship Obfuscation Approaches

As state-of-the-art automated authorship obfuscators we identified the top two systems (Potthast et al., 2018) from PAN, a shared CLEF task.[8] We also chose Mutant-X, a search based system presented in (Mahmood et al., 2019), which shows better performance than the PAN obfuscation systems. These are detailed next.

Document Simplification (Castro-Castro et al., 2017). This approach obfuscates by applying rule-based text simplifications on the input document. The process is as follows. 1) If the number of contractions in the document is greater than the number of expansions, then replace all contractions with expansions otherwise replace all expansions with contractions. 2) Simplify by removing parenthetical texts that do not contain any named entity, discourse markers or appositions. 3) Replace words with synonyms that haven't been already used in the text. We implement this approach and refer to it as DS-PAN17.

Style Neutralization (Karadzhov et al., 2017). This system is also a rule-based text obfuscator. First they calculate the average values for the whole corpus for stylometric features such as stopword to non stopword ratio, punctuation to word count ratio and average number of words per sentence. Next, they calculate the values of same stylomteric features for the input document. Finally, using text transformation rules (e.g., replace ! with !!, merge or split sentences etc.) they move the document's stylometric feature values towards the corpus averages. We evaluate this approach using the code provided by the authors and refer to it as SN-PAN16.

MUTANT-X (Mahmood et al., 2019) This system uses a genetic algorithm (GAs) in combination with an authorship attribution system to identify words that when changed would have the highest positive effect towards obfuscation. Text transformations are done using a sentiment preserving variation of Word2Vec (Yu et al., 2017). The authors present two versions: MUTANT-X writeprintsRFC, built using a traditional machine learning based authorship attribution system and MUTANT-X embeddingCNN, built using a deep learning based authorship attribution system. We evaluate MUTANT-X embeddingCNN using code made available by authors.

4.2 Data

We use the two data collections which were used by (Mahmood et al., 2019).

1) **Extended Brennan Greenstadt corpus**. This text corpus from (Brennan et al., 2012) contains 699 documents written by 45 unique authors. Documents are mainly academic in nature but they do not contain any citations and section headings and have under 500 words, as instructed by data collectors. We refer to this collection as EBG.

2) **Blog authorship corpus**. This text corpus which is from (Schler et al., 2006) contains more than 600,000 blogger.com blog posts written by 19,320 unique authors. These texts are more casual in nature and are written without any instructions. We refer to this collection as BLOG.

The authors in (Mahmood et al., 2019) tested their GA obfuscators using 49 EBG and 200 BLOG documents written by 10 authors.

4.2.1 Obfuscated versus Evaded Datasets

A document is considered obfuscated if it has been processed by an authorship obfuscation tool. Thus we processed each document in the 49 EBG and 200 BLOG sets using each of the three obfuscation systems in turn (DS-PAN17, SN-PAN16 and MUTANT-X embeddingCNN). This resulted in 127 obfuscated EBG documents and 571 obfuscated BLOG documents.[9]

An obfuscated document is viewed as an evaded document if it successfully evades authorship attribution. Using the writeprintsRFC authorship attributor (Mahmood et al., 2019) for this purpose, we process all obfuscated documents (127 and 571 for EBG and BLOG respectively). This resulted

[8] https://pan.webis.de/

[9] DS-PAN17, SN-PAN16 and MUTANT-X embeddingCNN respectively created 49, 49 and 29 obfuscated documents for EBG whereas for BLOG they created 200, 200 and 171 obfuscated documents. Lower number of documents generated by MUTANT-X is due to the fact that it only obfuscates documents that are correctly classified by the attribution classifier it uses.

Dataset	Train		Test	
	Original	Obfuscated/ Evaded	Original	Obfuscated/ Evaded
EBG obfuscated	431	80	268	47
EBG evaded	236	29	235	30
BLOG obfuscated	2000	292	1900	277
BLOG evaded	700	71	1000	113

Table 1: Number of original and obfuscated/evaded documents in train and test sets of each of the four datasets

in 59 evaded EBG documents (out of 127) and 184 evaded BLOG documents (out of 571).

Despite the type and extent of transformations made, the remaining obfuscated documents did not evade the attributor. The evaded documents are those where the modification strategy somehow crossed an implicit threshold for evading authorship attribution. Thus, we surmise that the evaded documents are likely to be relatively less smooth. It is noteworthy that evaded documents are the realistic goal from a privacy seeking user's perspective. Thus, in addition to assessing detection for obfuscated data, we also assess detection for evaded data for both (EBG and BLOG) obfuscated datasets.

4.2.2 Final datasets and cross validation

For each dataset, we use the original and obfuscated/evaded documents by 5 (out of 10) authors for training and the documents by the remaining 5 authors for testing. An important decision to make here is with regards to the distribution of original documents versus obfuscated (or evaded) ones in our training and testing set. We have no real world scenario to mimic in that we have not encountered any real world use of automated obfuscators and their outputs. Therefore we make the datasets under a reasonable assumption that original documents are in the vast majority. Thus we create train and test datasets with roughly 85% - 90% original documents. We do this by adding more original documents from each data collection to achieve these proportions. Overall, we ensure that each author's documents (of any version) are all either in the train or in the test set. Table 1 provides the size details for our obfuscated and evaded datasets.

4.3 Obfuscation Detection Methods

4.3.1 Methods from our architecture

We propose a total of 160 distinct architectures (see 3.2.4). Testing these for each of the four datasets, we conduct a total of 640 distinct obfuscation detection experiments. In addition, we explore three other methods; the first is inspired by recent research to detect synthetic text. The other two were used previously to detect manually obfuscated documents.

4.3.2 Other Methods

1) GLTR (Gehrmann et al., 2019). The authors present a tool to help humans distinguish between original and machine generated synthetic text. Their tool uses pretrained language models to extract word likelihoods and presents their plot to humans making the decision. Their best model uses GPT-2 117M language model to extract word ranks which are then put in 4 unequal range bins: 0-10, 10-100, 100-1000 and greater than 1000. We test the same method on our obfuscation detection problem. For obfuscation detection, we tested this feature set with linear SVM and RFC. Among these, linear SVM performed the best.

2) Character trigrams + KNN (Juola, 2012). This paper presents a stylometric approach to detect *manually* obfuscated documents. Firstly, they preprocess each document by unifying case and separating punctuation (e.g., !! becomes ! !). Then they represent each document using character trigrams. Finally the classification is performed using KNN with normalized cosine distance. We implement this approach to evaluate it against automated authorship obfuscaters.

3) Writeprints + SVM (Afroz et al., 2012). This paper also uses stylometric features and is also focused on detecting manual obfuscation. It is also similar to the approach in (Shahid et al., 2017), which uses stylometric features to perform spun document detection. Afroz et al. tested with 3 different feature sets of which writeprints was the best. Their writeprints feature set contains character related features (e.g., total characters, percentage of uppercase letters etc.), word related features (e.g., total words, frequency of large words etc.) and other features including frequencies of digits, special characters, function words etc. They use this feature set with SVM (using *poly* kernel) to

perform obfuscation detection. On our datasets, we found linear SVM to be working better than the polynomial one, so we report results with linear SVM. We implement this writeprints approach with linear SVM as our final detector.

5 Results

Summary trends: After averaging we find that for obfuscation detection, 25% of all 160 architectures achieve F1 score greater than 0.76, 50% achieve F1 score greater than 0.72 and a high 75% of them were able to achieve F1 score greater than 0.52.

Figure 2 summarizes the performances of all 160 different architectures across the four datasets. We see that obfuscation detection is easier in EBG than in BLOG with median EBG F1 scores being significantly better than median BLOG F1 scores (notches do not overlap (Krzywinski and Altman, 2014)). This can be explained by the fact that EBG contains scholarly articles that are relatively more consistent in their smoothness than blogs. This likely makes it easier to pick up on the difference in smoothness caused by obfuscated documents in EBG than in BLOG. We can also see that evaded documents achieve a higher maximum F1 score than obfuscated documents. This confirms our intuition presented in 4.2.1, that evaded documents are likely to be less smooth and therefore easier to detect than obfuscated documents. However, we also see that F1 scores for evaded datasets are less stable (greater box size) than obfuscated datasets. We believe that this is due to the fact that there are fewer documents in evaded datasets as compared to their respective obfuscated datasets (see Table 1).

Performance evaluation: In terms of architecture selection, instead of choosing randomly across 160 architectures, we make the following assumption. We assume that the adversary is knowledgeable about the various choices, tests these alternatives and employs the best configuration. Thus, we present results for the best models, based on F1 scores for obfuscation detection, achievable by the adversary (Table 2). Table 2 also presents results for the three additional methods presented in section 4.3.2. Our best BERT and GPT2 combinations outperform all other methods across each of the four datasets in F1 score and recall. Along with (GLTR + SVM) these achieve the best precision for the EBG datasets. In BLOG obfuscated, GLTR based method achieves the highest precision whereas in BLOG evaded both the GLTR based method and character trigrams method top the chart - however in each case with a sizeable penalty paid in recall and therefore in F1 score. In summary, we see that using the best of methods the adversary can detect evaded and obfuscated documents with F1 score of 0.77 or higher (average 0.87 across datasets) which indicates that the tested state-of-the-art obfuscators are far from stealthy.

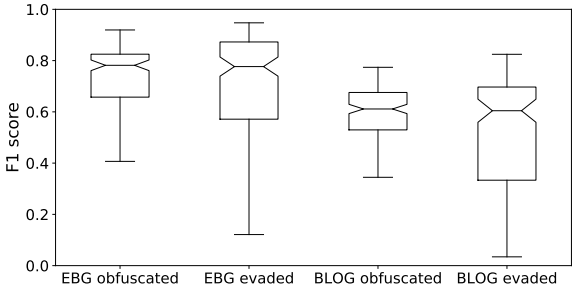

Figure 2: Notched box plots for obfuscation detection F1 scores using all 160 architectures for each dataset.

Dataset	Models	P	R	F1
EBG obfuscated	BERT LARGE + ranks + VGG-19 + RFC	1.00	0.85	**0.92**
	BERT LARGE + ranks + VGG-19 + SVM	0.98	0.83	**0.90**
	GLTR + SVM	1.00	0.70	0.83
	Writeprints + SVM	0.67	0.38	0.49
	Character trigrams + KNN	0.64	0.15	0.24
EBG evaded	BERT LARGE + probs + bins(0.010) + ANN	1.00	0.90	**0.95**
	BERT BASE + probs + VGG-19 + GNB	1.00	0.90	**0.95**
	GLTR + SVM	1.00	0.80	0.89
	Writeprints + SVM	0.79	0.63	0.70
	Character trigrams + KNN	1.00	0.17	0.29
BLOG obfuscated	BERT BASE + probs + VGG-19 + ANN	0.85	0.71	**0.77**
	BERT BASE + probs + VGG-19 + SVM	0.79	0.74	**0.77**
	GLTR + SVM	0.92	0.40	0.56
	Writeprints + SVM	0.71	0.41	0.52
	Character trigrams + KNN	0.41	0.50	0.45
BLOG evaded	GPT-2 345M + ranks + VGG-19 + GNB	0.82	0.83	**0.83**
	BERT BASE + probs + VGG-19 + ANN	0.79	0.81	**0.80**
	GLTR + SVM	0.86	0.55	0.67
	Writeprints + SVM	0.84	0.62	0.71
	Character trigrams + KNN	0.86	0.50	0.63

Table 2: Obfuscation detection results (P: precision, R: recall, F1: F1 score).

5.1 Detector Architecture Choices Analysis

Now we analyze the effect of different choices made within each of the three dimensions depicted in Figure 1. As mentioned earlier, for a privacy seeking user evading author attribution is more im-

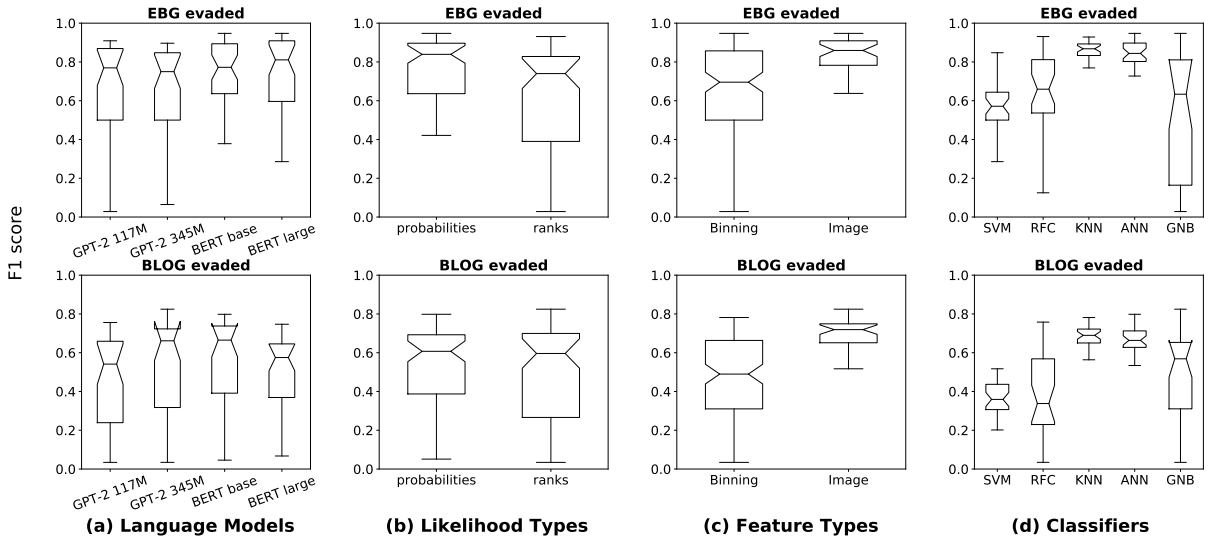

Figure 3: Notched box plots of F1 scores for all dimensions across the two evaded datasets. For each dataset every notched boxplot in (a) is generated from 40 experiments (experiments correspond to architectures), (b) is generated from 80 experiments, (c) is generated from 120 experiments for binning and 40 for image whereas (d) is generated from 32 different experimental combinations.

portant than just obfuscation. So, in this section we present architecture analysis results only for evaded datasets involving 320 experiments (160 each for EBG evaded and BLOG evaded).

5.1.1 Dimension 1: Language model & output type

Figure 3 (a) presents notched box plots comparing distribution of F1 scores achieved by language models across both datasets. In EBG evaded, BERT language models achieve higher maximum F1 score (0.95) than GPT-2 (0.90 - 0.91). On the other hand, in BLOG evaded, GPT-2 345M achieves higher maximum F1 score (0.83) than others (0.75 - 0.80). Relatively, BERT shows greater consistency in F1 score (box size) than GPT-2 in both datasets. We believe that the bidirectional nature of BERT helps in capturing context and consequently smoothness better than GPT-2 which is uni-directional.

While the difference in maximum F1 score between ranks and probabilities is slight for each dataset (Figure 3 (b)) box sizes show the spread in F1 scores is smaller with probabilities than with ranks. Upon further investigation, we find that experiments which use probabilities with image based features have an inter-quartile range of 0.05 and 0.1 for EBG and BLOG respectively whereas for experiments using probabilities with binning based features, this range is 0.32 for both datasets. On the other hand, inter-quartile range for experiments using ranks with image based features is 0.08 and 0.05 for EBG and BLOG whereas for experiments using ranks with binning based features, this range is 0.49 and 0.42 respectively. This shows that for both datasets, greater variation in F1 scores for ranks as compared to probabilities is caused by binning based features. We believe that binning ranks with fixed bin sizes (10, 50, 100) is less stable for both BERT and GPT-2 which have different limits of ranks - this could account for the larger inter-quartile range using ranks.

5.1.2 Dimension 2: Feature type

The box sizes in Figure 3 (c) show that image based features exhibit strikingly greater stability in F1 scores than binning based features. Image based features also achieve significantly higher median F1 score than with binning for both datasets. This can in part be explained by the observation stated earlier that some bin size choices tested perform much worse than others because of not being fine-tuned. There is no difference between feature types in maximum F1 score for EBG whereas in BLOG, image based feature achieve somewhat higher maximum F1 score (0.83) than binning based features (0.78). We believe that the reason why image based features work so well is that VGG-19, the image model we use to extract features, is powerful enough to recognize the slopes in plots which represent the smoothness in our case.

5.1.3 Dimension 3: Classifier

Figure 3 (d), shows that for EBG, ANN and GNB achieve higher maximum F1 score (0.95), whereas for BLOG, GNB achieve higher maximum F1 score (0.83). KNN and ANN consistently achieve far more stable F1 scores than other classification methods. In both datasets, KNN achieves significantly higher median F1 score than other classification methods. ANN also follows the same pattern with the exception of GNB in BLOG evaded. We believe that the reason why KNN and ANN achieve relatively high and stable performance is in their nature of being able to adapt to diverse and complex feature spaces.

5.2 Takeaway

In summary we conclude that BERT with probabilities is a good choice for dimension 1. (We remind the reader that in contrast, in the area of synthetic text detection (Gehrmann et al., 2019) GPT-2 had the edge over BERT). Image based features are a clear winner in dimension 2 while KNN and ANN are the best candidates for dimension 3. Key to note as well is that the top performing architectures in Table 2 differ across datasets indicating the need for dataset specific choices.

5.3 Insights

Figure 4 validates our intuition from Section 3 that the text generated by obfuscators is less smooth than the original text. Using EBG obfuscated dataset and BERT BASE for illustration, we first sort words in a document by estimated probability and plot average probability at each rank. The steeper the fall in the curve, the lower the smoothness of text. This plot shows that original documents are generally more smooth than obfuscated documents. The average detection error rates (Mutant-X embeddingCNN: 0.72, SN-PAN16: 0.48, and DS-PAN17: 0.07) are also consistent with the plot. These results show that Mutant-X is the most stealthy obfuscator while DS-PAN17 is the least stealthy obfuscator.

6 Conclusion

In this paper, we showed that the state-of-the-art authorship obfuscation methods are not stealthy. We showed that the degradation in text smoothness caused by authorship obfuscators allow a detector to distinguish between obfuscated documents and original documents. Our proposed obfuscation detectors were effective at classifying obfuscated and evaded documents (F1 score as high as 0.92 and 0.95, respectively). Our findings point to future research opportunities to build stealthy authorship obfuscation methods. We suggest that obfuscation methods should strive to preserve text smoothness in addition to semantics.

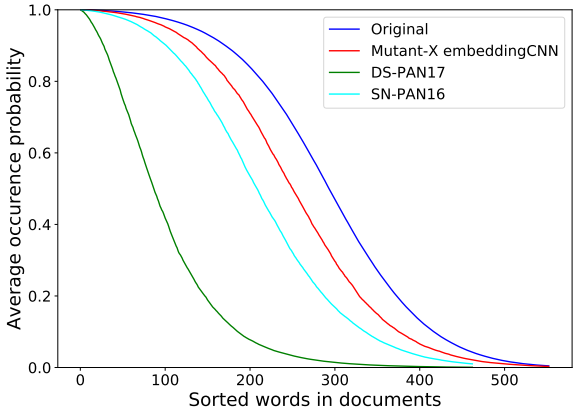

Figure 4: Comparison between different obfuscators and original documents on the basis of average sorted probabilities extracted by BERT BASE for EBG obfuscated dataset.

References

2018. PAN @ CLEF 2018 - Author Obfuscation. https://pan.webis.de/clef18/pan18-web/author-obfuscation.html.

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

Sadia Afroz, Michael Brennan, and Rachel Greenstadt. 2012. Detecting hoaxes, frauds, and deception in writing style online. In *2012 IEEE Symposium on Security and Privacy*, pages 461–475. IEEE.

Mishari Almishari, Ekin Oguz, and Gene Tsudik. 2014. Fighting Authorship Linkability with Crowdsourcing. In *ACM Conference on Online Social Networks (COSN)*.

Anonymous. 2018. I'm an Amazon Employee. My Company Shouldn't Sell Facial Recognition Tech to Police.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or Fake? Learning to Discriminate Machine from Human Generated Text. *arXiv preprint arXiv:1906.03351*.

Janek Bevendorff, Martin Potthast, Matthias Hagen, and Benno Stein. 2019. Heuristic Authorship Obfuscation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1098–1108.

Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):12.

Daniel Castro-Castro, Reynier Ortega Bueno, and Rafael Munoz. 2017. Author Masking by Sentence Transformation. In *Notebook for PAN at CLEF*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Emmery, Enrique Manjavacas, and Grzegorz Chrupała. 2018. Style Obfuscation by Invariance. *27th International Conference on Computational Linguistics*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. pages 111–116.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. CNN architectures for large-scale audio classification. In *2017 IEEE international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE.

Patrick Juola. 2012. Detecting stylistic deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 91–96. Association for Computational Linguistics.

Georgi Karadzhov, Tsvetomila Mihaylova, Yasen Kiprov, Georgi Georgiev, Ivan Koychev, and Preslav Nakov. 2017. The case for being average: A mediocrity approach to style masking and author obfuscation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 173–185. Springer.

Yashwant Keswani, Harsh Trivedi, Parth Mehta, and Prasenjit Majumder. 2016. Author Masking through Translation. In *Notebook for PAN at CLEF 2016*, pages 890–894.

Martin Krzywinski and Naomi Altman. 2014. Points of significance: visualizing samples with box plots.

Asad Mahmood, Faizan Ahmad, Zubair Shafiq, Padmini Srinivasan, and Fareed Zaffar. 2019. A Girl Has No Name: Automated Authorship Obfuscation using Mutant-X. *Proceedings on Privacy Enhancing Technologies*, 2019(4):54–71.

Muharram Mansoorizadeh, Taher Rahgooy, Mohammad Aminiyan, and Mahdy Eskandari. 2016. Author obfuscation using WordNet and language models. In *Notebook for PAN at CLEF 2016*.

Andrew WE McDonald, Sadia Afroz, Aylin Caliskan, Ariel Stolerman, and Rachel Greenstadt. 2012. Use fewer instances of the letter "i": Toward writing style anonymization. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 299–318. Springer.

Andrew W.E. McDonald, Jeffrey Ulman, Marc Barrowclift, and Rachel Greenstadt. 2013. Anonymouth Revamped: Getting Closer to Stylometric Anonymity. In *PETools: Workshop on Privacy Enhancing Tools*, volume 20.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314. IEEE.

Rebekah Overdorf and Rachel Greenstadt. 2016. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2016(3):155–171.

Martin Potthast, Felix Schremmer, Matthias Hagen, and Benno Stein. 2018. Overview of the author obfuscation task at pan 2018: A new approach to measuring safety. In *CLEF (Working Notes)*.

Schremmer Potthast and Stein Hagen. 2018. Overview of the Author Obfuscation Task at PAN 2018: A New Approach to Measuring Safety. In *Notebook for PAN at CLEF 2018*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 6, pages 199–205.

Usman Shahid, Shehroze Farooqi, Raza Ahmad, Zubair Shafiq, Padmini Srinivasan, and Fareed Zaffar. 2017. Accurate detection of automatically spun content via stylometric analysis. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 425–434. IEEE.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4NT: author attribute anonymity by adversarial training of neural machine translation. In *27th*

USENIX Security Symposium (USENIX Security 18), pages 1633–1650.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

NY Times. 2018. I am part of the resistance inside the trump administration. *NY Times. Retrieved from https://www. nytimes. com/2018/09/05/.../trump-white-house-anonymous-resistance. html*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings using intensity scores for sentiment analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):671–681.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Conference on Neural Information Processing Systems (NeurIPS)*.