

# Enhancing Adversarial Authorship Verification with Data Augmentation

Discussion Paper

Silvia Corbara<sup>1,2,\*</sup>, Alejandro Moreo<sup>2,†</sup>

<sup>1</sup>Scuola Normale Superiore, 56126 Pisa, Italy

<sup>2</sup>Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124 Pisa, Italy

## Abstract

It has been shown that many Authorship Identification systems are vulnerable to adversarial attacks, where an author actively tries to fool the classifier. We propose to tackle the adversarial Authorship Verification task by augmenting the training set with synthetic textual examples. In this ongoing study, we present preliminary results using two learning algorithms (SVM and Neural Network), and two generation strategies (based on language modeling and GAN training) for two generator models, on three datasets. We empirically show that data augmentation may help improve the performance of the classifier in an adversarial setup.

## Keywords

Authorship Verification, Data augmentation, Text classification

## 1. Introduction

*Authorship Identification* (AId) is the branch of *Authorship Analysis* concerned with uncovering the true identity of the author of a written document whose paternity is unknown or debated. *Authorship Verification* (AV) is one of the main tasks of AId in which, given a single candidate author  $A$  and a target document  $d$ , the goal is to infer whether  $A$  is the real author of  $d$  or not [1]. AV is thus framed as a binary text classification task with  $A$  and  $\bar{A}$  as the possible classes.

However, the performance of the classifier might be negatively affected when an “adversary” is at play, i.e., when an agent (a human or a computer) actively tries to mislead the classifier, either by concealing their own writing style or by imitating someone else’s [2]. Our cultural heritage is indeed filled with countless examples of presumed forgeries or false appropriations [3, 4, 5, 6]. To cap it all, modern Neural Networks (NNs) are constantly improving their ability to autonomously generate convincing human-like pieces of text that can be exploited as fake news or propaganda [7, 8]. Indeed, it has been conclusively shown that many AId systems can be easily fooled in adversarial contexts [2, 9].

In this ongoing work, we investigate ways for improving the performance of an AV classifier by augmenting the training set with synthetically generated examples. The intuition is that, if a classifier has been exposed to textual examples that mimic an adversarial setting during training, it may develop resiliency toward cases of forgery. In this short paper, we present some preliminary results of our study. All the experiments are implemented in Python; the code to reproduce our experiments is available at: [https://github.com/silvia-cor/Authorship\\_DataAugmentation](https://github.com/silvia-cor/Authorship_DataAugmentation).

---

IIR2023: 13th Italian Information Retrieval Workshop, 8th - 9th June 2023, Pisa, Italy

\*Corresponding author.


†These authors contributed equally.

✉ [silvia.corbara@sns.it](mailto:silvia.corbara@sns.it) (S. Corbara); [alejandro.moreo@isti.cnr.it](mailto:alejandro.moreo@isti.cnr.it) (A. Moreo)

🆔 0000-0002-5284-1771 (S. Corbara); 0000-0002-0377-1025 (A. Moreo)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

## 2. Related work

The annual PAN shared tasks [10] offer a comprehensive overview of the most recent trends in AId. In particular, SVMs have become a standard learning algorithm for these tasks, outperforming many other methods [11]. However, deep NN methods are becoming more and more prevalent [12, 13, 10].

The use of data augmentation for improving classification performance is not a novelty. Researchers have explored various techniques for generating synthetic textual examples, such as the random combination of real texts [14, 15], the random substitution of words with synonyms [16], or by employing large language models to generate texts [17]. Similarly, adversarial examples (i.e., examples specifically generated to fool the model [18]) have been extensively used to improve the training phase of classifiers in many text classification tasks. For example, Zhai et al. [19] feed the learning algorithm with (real) texts that have been purposefully obfuscated, in order to improve the robustness of the classifier. Unlike these projects, we do not modify pre-existing texts, but instead create new ad-hoc examples.

The work by Hatua et al. [20] bears some similarities with our methodology: in their experiments tackling the fact-checking task, data augmentation with a GAN-based generator proved useful. Note that, unlike this work, we cannot employ the generated examples as synthetic positive instances, but rather as particularly difficult instances for the negative class. Otherwise, the AV classifier would learn to label fraudulent instances as if written by the author of interest.

## 3. Experiments

### 3.1. Datasets

The datasets we consider are described below. For each dataset, we split each document into non-overlapping chunks of 100 tokens (words and punctuation symbols); chunks with less than 25 words are discarded. We exclude authors with less than 10 chunks in the training set.

- **TweepFake.** This dataset was created and made publicly available<sup>1</sup> by Fagni et al. [7]. The dataset consists of tweets from 17 human accounts and from 23 bots, each one imitating one of the human accounts. We use this dataset as a reasonable proxy of the action of a forger. We use the training, validation and test partitions provided in the dataset, but we remove all documents produced by the bots from the training set, thus setting our AV problem as an open-set one.
- **RJ.** The Riddell-Juola Corpus was created and made publicly available<sup>2</sup> by Riddell et al. [21]; we focus on the “Obfuscation” setting. Participants on the Amazon Mechanical Turk platform were requested to (i) upload some past textual examples of their own production, and (ii) write a new short essay, where some random participants were asked to try and obfuscate their writing style. We randomly select 10 authors, and split all their chunks gathered from step (i) into a training set (90%) and a validation set (10%). We use the chunks gathered from step (ii) from all authors as the test set, thus simulating an open-set scenario. This dataset is representative of cases where the authors actively try to mask their writing.
- **Victoria.** This dataset was created and made publicly available<sup>3</sup> by Gungor [22]. It consists of books by American or English 18th-19th century novelists. We use these documents as examples of literary production, where no author is presumably trying to imitate someone else’s style, nor conceal their own identity. We limit the dataset to 5 randomly selected authors with at most 1,000 chunks each. We devote 90% of the documents for training and the remaining 10% for test. The training set is further split into a (proper) training set consisting of 90% of the documents, and a validation set that contains the remaining 10%. This setting is representative of an AV closed-set problem in which there are few authors, each with an abundant production.

<sup>1</sup>A limited version is available on Kaggle at: <https://www.kaggle.com/datasets/mtesconi/twitter-deep-fake-text>.

<sup>2</sup>Available on the Reproducible Authorship Attribution Benchmark Tasks (RAABT) on Zenodo: <https://zenodo.org/record/5213898#.YuuaNdJBzys>

<sup>3</sup>Available at: <https://archive.ics.uci.edu/ml/datasets/Victorian+Era+Authorship+Attribution>.

### 3.2. Generators

We experiment with two architectures for generating adversarial examples. The first one is Distil-GPT2 [23] (hereafter simply denoted as GPT), a distilled variant of OpenAI GPT-2, made available as part of the Huggingface’s transformers library.<sup>4</sup> The second one, denoted  $\text{TRA}_{1\text{h}}$ , is based on the TransformerEncoder module by Pytorch<sup>5</sup> [24], and operates with one-hot vectors. The inputs are processed by a linear layer (without bias) that converts the sparse one-hot vectors into dense embeddings.

We experiment with two different training strategies for the generators:

- **Language Model Training (LM).** The model is trained to predict the next word given a sequence. More formally, given a real sentence  $[w_1, w_2, \dots, w_t]$  of  $t$  tokens, the model aims to maximize the conditional probability  $P(w_t | w_1, w_2, \dots, w_{t-1})$ . In our case, following [25], we train the model only using the written examples of  $A$  (our author of interest). The generator thus is expected to act as an imitator of  $A$ ’s writing style.
- **Generative Adversarial Network Training (GAN).** A Generative Adversarial Network (GAN) [26] has two components: a Generator ( $G$ ), that produces fake new examples, and a Discriminator ( $D$ ), that labels examples either as “real” (i.e., coming from the real-world distribution) or “fake” (i.e., produced by  $G$ ). Both components play a min-max game, where  $D$  tries to correctly discover the forgeries issued by  $G$ , while at the same time  $G$  tries to produce examples that manage to fool  $D$ . We explore various configurations inspired by GANs, in which we vary the generator and the discriminator. In particular, we use one of the NN classifiers ( $\text{NN}_{\text{gpt}}$  or  $\text{NN}_{1\text{h}}$ ) explained in Section 3.3 as our discriminator  $D$ , while we use GPT or  $\text{TRA}_{1\text{h}}$  as our generator  $G$ . The generator is thus trained to generate examples that are supposed to be particularly difficult for the classifier.

In order to generate sequences of tokens, one typically resorts to the argmax of the posterior distribution over the token vocabulary. However, the argmax operator breaks the flow of the gradient throughout the network, thus obstructing the GAN training. In order to avoid this, we either generate sequences of hidden states in the case of GPT, or apply the Gumbel-Softmax [27] in the case of  $\text{TRA}_{1\text{h}}$ <sup>6</sup>.

### 3.3. Learning algorithms

We consider two different learning algorithms, Support Vector Machine (SVM) and Convolutional Neural Networks (NN) as methods for training the AV classifier.

For SVM, we employ the SVC implementation from the scikit-learn package<sup>7</sup>, with hyper-parameter optimization. We employ a combination of features including the normalized relative frequency of function words and the POS-tags, and the word lengths; we call these features BaseFeatures. We also extract all character  $n$ -grams (with  $n$  in the range  $[1, 3]$ ) and compute their TFIDF weights; we retain only the 10% most discriminating ones according to their  $\chi^2$  value.

For NN, we develop an architecture with two parallel branches. In one branch, when the input is a real example, words are converted to one-hot vectors that are simply embedded through a linear layer (without bias); when the input is a fake example, the embeddings are either generated by feed-forwarding the “quasi” one-hot vectors resulting from the Gumbel-Softmax in the case of  $\text{TRA}_{1\text{h}}$  (we call this setting  $\text{NN}_{1\text{h}}$ ), or are instead directly taken from the hidden states of GPT (we call this setting  $\text{NN}_{\text{gpt}}$ ). The rest of the branch is composed of two parallel convolutional blocks, each with two convolutional layers with kernel size equal to 3 and 5 respectively. The other branch is made of two linear layers and receives the same BaseFeatures as the SVM classifier as input; this branch is simply skipped when the inputs are taken from hidden states, and thus cannot be converted into sentences. The outputs of

<sup>4</sup>Documentation available at: <https://huggingface.co/distilgpt2>.

<sup>5</sup>Documentation available at: <https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html>.

<sup>6</sup>This means that the examples we generate at training time are not comprehensible for a human. However, note this is not a problem since they are solely intended to imitate the input format that the discriminator receives as input in the GAN training. Viceversa, the trained generators do indeed produce examples that can be human-readable.

<sup>7</sup>Documentation available at: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

**Table 1**

Results of the experiments. The best model for each learning algorithm is highlighted in **bold**, the overall best model is in **underlined**, and the overall worst model is in *italic*.

	TweepFake					RJ-obf					Victoria				
	$F_1$	$\Delta\%$	Acc	$\Delta\%$	M	$F_1$	$\Delta\%$	Acc	$\Delta\%$	M	$F_1$	$\Delta\%$	Acc	$\Delta\%$	M
SVM	.320		<i>.870</i>			.200		.932			.750		.903		
SVM+GPT <sup>LM</sup>	.323	+ 0.98	<b>.956</b>	+ 9.97	✓	<b>.217</b>	+ 8.35	<b>.956</b>	+ 2.53	✓	.732	- 2.37	.888	- 1.73	✓
SVM+GPT <sup>GAN</sup>	<b>.333</b>	+ 3.98	.946	+ 8.75	✓	.200	—	.949	+ 1.80	✓	<b>.773</b>	+ 3.07	<b>.910</b>	+ 0.80	✗
SVM+TRA <sup>LM</sup> <sub>1h</sub>	.310	- 3.08	.952	+ 9.51	✓	.210	+ 4.75	.948	+ 1.66	✓	.762	+ 1.60	.902	- 0.09	✗
SVM+TRA <sup>GAN</sup> <sub>1h</sub>	.324	+ 1.35	.943	+ 8.40	✓	<i>.200</i>	—	.942	+ 1.06	✓	.753	+ 0.35	.906	+ 0.27	✗
NN <sub>gpt</sub>	.302		.962			.143		.904			.721		.898		
NN <sub>gpt</sub> +GPT <sup>LM</sup>	.359	+ 18.88	<b>.969</b>	+ 0.75	✓	.125	- 12.77	.908	+ 0.48	✗	.684	- 5.13	.879	- 2.09	✓
NN <sub>gpt</sub> +GPT <sup>GAN</sup>	<b>.371</b>	+ 23.17	.961	- 0.13	✗	<b>.227</b>	+ 58.13	<b>.919</b>	+ 1.71	✓	<b>.729</b>	+ 1.11	<b>.889</b>	- 1.02	✗
NN <sub>1h</sub>	.353		.963			.127		<b>.931</b>			<b>.739</b>		.894		
NN <sub>1h</sub> +TRA <sup>LM</sup> <sub>1h</sub>	.351	- 0.70	<b>.967</b>	+ 0.42	✓	.117	- 8.33	.875	- 6.09	✓	.722	- 2.27	.892	- 0.27	✗
NN <sub>1h</sub> +TRA <sup>GAN</sup> <sub>1h</sub>	<b>.363</b>	+ 2.68	.966	+ 0.37	✓	<b>.226</b>	+ 77.44	.928	- 0.34	✗	.730	- 1.14	<b>.901</b>	+ 0.76	✗

the two branches are finally concatenated and fed into two linear layers, that produce the posterior probabilities for the classes  $A$  and  $\bar{A}$ . For the training process, we employ the AdamW optimizer [28] with a learning rate of 0.001, a batch size of 32, and Cross Entropy as the loss function.

### 3.4. Experimental protocol and results

Given a dataset, we sequentially take each author in the training set as the author of interest  $A$  and all other authors as  $\bar{A}$ . We focus on evaluating the improvement obtained by augmenting the training data fed to the classifier (SVM, NN<sub>gpt</sub>, and NN<sub>1h</sub>). We first evaluate the classifier trained without any augmentation, and use it as baseline to assess the contribution of the generated samples when added to the training as negative instances. We define the classifiers trained via data augmentation with the nomenclature  $C + G^T$ , where  $C$  is the classifier,  $G$  is the generator and  $T$  is the generator training strategy.

Each time we generate examples (this happens for each epoch in the GAN training, and for generating the final data augmentation when the generator training is over), we produce  $\min\{5 \times n, 500\}$  examples, where  $n$  is the number of training examples by  $A$ . The tokens are generated by prompting the first 5 original tokens from a randomly chosen example by  $A$  until the length of the original text is reached. We use vanilla accuracy and  $F_1$  as our evaluation metrics, and report averaged results across all AV experiments. For each evaluation metric, we also report the relative improvement ( $\Delta\%$ ) with respect to the classifiers without data augmentation. We also carry out tests of statistical significance for the performance difference ( $M$ ) via the McNemar’s paired non-parametric test [29] at a confidence value of 0.05. The results are reported in Table 1.

Although it is not possible to identify one single approach that positively affects the classification of every dataset, the results are indeed encouraging. Data augmentation leads to a significant improvement in performance (both in accuracy and  $F_1$ ) in 10 cases out of 24, while it leads to a significant detriment for both metrics only in 3 cases. The best overall result is always attained by the GPT-based augmentation. Understandably, the effect on the Victoria dataset is less pronounced, likely due to the fact that this dataset does not contain forgers, and since the number of documents per author is fairly high.

## 4. Conclusion and Future Work

In this study, we present preliminary experiments regarding the idea of improving the performance of an Authorship Verification system by enhancing the training set with additional examples purposefully generated to simulate an adversary. Despite not being conclusive, the results we have obtained are promising, and encourage us to continue this research.

In future work, we plan to extend the experimentation to other datasets representative of different settings, and explore the suitability of simpler architectures and different training strategies [30] that might be better attuned to the typically limited number of available examples.

## References

- [1] E. Stamatatos, Authorship verification: A review of recent advances, *Research in Computing Science* 123 (2016) 9–25.
- [2] M. Brennan, S. Afroz, R. Greenstadt, Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity, *ACM Transactions on Information and System Security (TISSEC)* 15 (2012) 1–22.
- [3] S. Corbara, A. Moreo, F. Sebastiani, M. Tavoni, The Epistle to Cangrande through the lens of computational authorship verification, in: *Proceedings of the 1st International Workshop on Pattern Recognition for Cultural Heritage (PatReCH 2019)*, Trento, IT, 2019, pp. 148–158. doi:10.1007/978-3-030-30754-7\_15.
- [4] R. McCarthy, J. O’Sullivan, Who wrote *Wuthering Heights*?, *Digital Scholarship in the Humanities* 36 (2021) 383–391.
- [5] J. Savoy, Authorship of Pauline epistles revisited, *Journal of the Association for Information Science and Technology* 70 (2019) 1089–1097.
- [6] R. Vainio, R. Välimäki, A. Hella, M. Kaartinen, T. Immonen, A. Vesanto, F. Ginter, Reconsidering authorship in the Ciceronian corpus through computational authorship attribution, *Ciceroniana On Line* 3 (2019).
- [7] T. Fagni, F. Falchi, M. Gambini, A. Martella, M. Tesconi, TweepFake: About detecting deepfake tweets, *Plos one* 16 (2021) e0251415.
- [8] J. Salminen, C. Kandpal, A. M. Kamel, S.-g. Jung, B. J. Jansen, Creating and detecting fake reviews of online products, *Journal of Retailing and Consumer Services* 64 (2022).
- [9] M. Potthast, M. Hagen, B. Stein, Author obfuscation: Attacking the state of the art in authorship verification, *CLEF (Working Notes)* (2016) 716–749.
- [10] E. Stamatatos, M. Kestemont, K. Kredens, P. Pezik, A. Heini, J. Bevendorff, B. Stein, M. Potthast, Overview of the authorship verification task at PAN 2022, in: *CEUR workshop proceedings*, volume 3180, 2022, pp. 2301–2313.
- [11] R. Zheng, J. Li, H. Chen, Z. Huang, A framework for authorship identification of online messages: Writing-style features and classification techniques, *Journal of the American society for information science and technology* 57 (2006) 378–393.
- [12] J. Bevendorff, B. Ghanem, A. Giachanou, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. M. R. Pardo, P. Rosso, G. Specht, E. Stamatatos, B. Stein, M. Wiegmann, E. Zangerle, Overview of PAN 2020: Authorship verification, celebrity profiling, profiling fake news spreaders on Twitter, and style change detection, in: A. Arampatzis, E. Kanoulas, T. Tsirikika, S. Vrochidis, H. Joho, C. Lioma, C. Eickhoff, A. Névéol, L. Cappellato, N. Ferro (Eds.), *Proceedings of the Experimental IR Meets Multilinguality, Multimodality, and Interaction - 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22-25, 2020*, volume 12260 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 372–383.
- [13] J. Bevendorff, B. Chulvi, G. L. D. la Peña Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, E. Zangerle, Overview of PAN 2021: Authorship verification, profiling hate speech spreaders on twitter, and style change detection, in: K. S. Candan, B. Ionescu, L. Goeuriot, B. Larsen, H. Müller, A. Joly, M. Maistro, F. Piroi, G. Faggioli, N. Ferro (Eds.), *Proceedings of the Experimental IR Meets Multilinguality, Multimodality, and Interaction - 12th International Conference of the CLEF Association, CLEF 2021, Virtual Event, September 21-24, 2021*, volume 12880 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 419–431.
- [14] A. Theophilo, R. Giot, A. Rocha, Authorship attribution of social media messages, *IEEE Transactions on Computational Social Systems* (2021).
- [15] B. Boenninghoff, R. M. Nickel, S. Zeiler, D. Kolossa, Similarity learning for authorship verification in social media, in: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 2457–2461.

- [16] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, volume 1 of *NIPS'15*, MIT Press, Cambridge, MA, USA, 2015, pp. 649–657.
- [17] S. Kobayashi, Contextual augmentation: Data augmentation by words with paradigmatic relations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 2, ACL, 2018, pp. 452–457. URL: <https://aclanthology.org/N18-2072>. doi:10.18653/v1/N18-2072.
- [18] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv e-prints (2014) arXiv:1412.6572. arXiv:1412.6572.
- [19] W. Zhai, J. Rusert, Z. Shafiq, P. Srinivasan, A girl has a name, and it's... adversarial authorship attribution for deobfuscation, arXiv preprint arXiv:2203.11849 (2022).
- [20] A. Hatua, A. M. Mukherjee, R. Verma, On the feasibility of using GANs for claim verification-experiments and analysis, in: Proceedings of the 2021 Workshop on Reducing Online Misinformation Through Credible Information Retrieval, 2021.
- [21] A. Riddell, H. Wang, P. Juola, A call for clarity in contemporary authorship attribution evaluation, in: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), 2021, pp. 1174–1179.
- [22] A. Gungor, Benchmarking Authorship Attribution Techniques Using Over a Thousand Books by Fifty Victorian Era Novelists, Ph.D. thesis, Purdue University, 2018.
- [23] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter, in: NeurIPS EMC<sup>2</sup> Workshop, 2019.
- [24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5998–6008. URL: <https://papers.nips.cc/paper/7181-attention-is-all-you-need>.
- [25] K. Jones, J. R. C. Nurse, S. Li, Are you Robert or RoBERTa? Deceiving online authorship attribution models using neural text generators, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 16, 2022, pp. 429–440.
- [26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (2020) 139–144.
- [27] M. J. Kusner, J. M. Hernández-Lobato, GANs for sequences of discrete elements with the Gumbel-softmax distribution, arXiv e-prints (2016) arXiv:1611.
- [28] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2018.
- [29] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (1947) 153–157.
- [30] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.