The Pennsylvania State University
The Graduate School

REVERSE TURING TEST IN THE AGE OF DEEPFAKE TEXTS

A Dissertation in
Information Sciences and Technology
by
Adaku Uchendu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2023

The dissertation of Adaku Uchendu was reviewed and approved by the following:

Abstract

As Artificial Intelligent (AI) technologies become ubiquitous, humans will have to contend with many benefits and disadvantages of these advancements. Particularly, in recent years, Natural Language Generation (NLG) methods have massively improved in the generation of well-written human-quality texts, leading to sophisticated Large-scale Language Models (LLMs) such as ChatGPT, GPT-4, and LLaMA. Naturally, these newer LLMs can generate texts that can be easily misconstrued as human-written, thus causing the problem of "deepfake texts" and exaggerating and exacerbating the potential for misuse (e.g., the generation of misinformation at scale).

In this dissertation, thus, we strive to understand the phenomenon surrounding deepfake texts better and develop solutions to tackle the problem. We first frame the detection of deepfake texts using the Reverse Turing Test (RTT) concept, which entails automatically distinguishing AI-generated texts from human-written ones. Then, we address 3 Research Questions (RQs): (1) Can one detect subtle linguistic differences between AI-generated texts and human-written ones? (2) Can one build an accurate deepfake text detector to distinguish between human vs. deepfake text authorship? (3) How can one improve human performance in detecting deepfake texts?

RQ1 is addressed by building an interpretable linguistic model to capture the subtle linguistic patterns that distinguish human-written from deepfake texts. RQ2 is addressed by several types of deepfake text detectors—e.g., stylometric, deep learning based, and a hybrid model (i.e., an ensemble of a Transformer-based model and Topological Data Analysis (TDA) techniques). RQ3 is addressed by studying how English experts vs. English non-experts perform at detecting deepfake texts in an individual and collaborative setting.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work was in part supported by NSF awards #1742702, #1820609, #1915801, #1934782, #1909702, #2114824, and #2131144. Additionally, I acknowledge and thank the graduate school at Penn State for awarding me with the Bunton-Waller fellowship. Furthermore, I would like to thank the NSF in collaboration with the Department of Homeland Security (DHS) and the Office of Personnel Management (OPM) for supporting me with the NSF SFS scholarship for 3 years. Lastly, I would also like to thank the Alfred P. Sloan Foundation for also supporting me in my graduate school career with a generous scholarship. Finally, the findings and conclusion do not necessarily reflect the view of the funding agencies.

Dedication

This dissertation is dedicated to my family - Chima, Nkechi, and Uchendu Uchendu.

Chapter 1 | Introduction

1.1 Background

The recent increasing ubiquity of Artificial Intelligent technologies has many benefits and drawbacks. One significant development in AI, specifically the Natural Language Generation (NLG) field is the creation of Large Language Models (LLMs) which are able to generate authentic-looking human-like texts. LLMs are probabilistic models trained on massive amounts of data. These models can be used for downstream tasks such as text classification, generation, summarization, etc. However, the LLMs, we discuss in this dissertation are text-generative LLMs capable of generating long-coherent human-like texts. We call these texts generated by text-generative LLMs, *deepfake texts*. Deepfakes can be defined as content generated by an AI model or algorithm. This content can either be images, videos, or texts and in our case, we refer deepfake texts are AI-generated texts. Additionally, the word *deepfake* in this instance does not consider the factual grounding of the AI-generated texts, so just because a piece of text is deepfake, does not mean it is fake or real news in this Dissertation.

These LLMs (or deepfake text-generators) can be exploited malicious to generate realistic misinformation at scale due to their impressive generative qualities. Some of these sophisticated text generators include ChatGPT, GPT-4, LLaMA, Google's Switch, etc. Therefore, it is very imperative that we build solutions to distinguish deepfake texts from human-written texts. Before we discuss these solutions, we must first describe the *Turing Test* [4] principle, in which our task bears a strong similarity. *Turing Test* is a test defined by Alan Turing in the 1950s, administered by a human to test the intelligence of a machine. In this test, the human is in a double-blind scenario where they might be speaking to a human or a machine. Based on the response of the dialogue participants, the human judge has to guess who they are speaking to. If a machine shows intelligent behavior usually attributed to a human, and thus classified as human, the machine has passed the *Turing Test*. For the purposes of this work, we aim to build Machine learning and Deep learning models to automatically distinguish between deepfake and human-written texts, to minimize the changes of the machine, in this case, LLMs passing the *Turing Test*. Therefore, in our task, a computer algorithm administers the *Turing Test* (TT). This automatic *Turing Test* is known as **Reverse Turing Test** (RTT). We also study a variation of the *Turing Test* problem, called *Authorship Attribution* (AA). First, ***Turing Test*** is a binary classification problem where we ask *is the given text T, authored by a human or deepfake text-generator?* and ***Authorship Attribution*** is a multinomial classification problem where we ask *given a text T and k candidate deepfake text-generators, can we single out the generator (among k choices) that generated T?*

Given the obvious security risks posed by deepfake text-generators, nuanced solutions must be proposed for this nuanced problem. Therefore, we propose novel solutions to solve this problem, which address 3 Research Questions (RQs): (1) Can we detect subtle linguistic differences of between deepfake texts and human-written ones? (2) Can we build an accurate deepfake text detector to attribute human vs. deepfake text authorship? (3) Can we improve human performance in detecting deepfake texts?

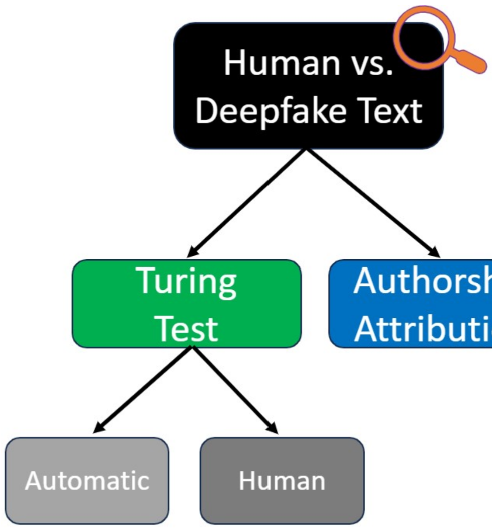


Figure 1.1: Flow chart of the 3 Research Questions (RQs) addressed in this Dissertation.

RQ1 is addressed by building an interpretable linguistic model to capture the subtle linguistic patterns that distinguish human-written from deepfake texts. RQ2 is addressed by building several types of automatic deepfake text detectors, including an ensemble of Transformer-based model - RoBERTa and Topological Data Analysis (TDA) techniques. RQ3 is addressed by studying how English experts vs. English non-experts perform the task of deepfake text detection in an individual and collaborative setting. See Figure 6.1 of the flow chart of 3 RQs addressed in this Dissertation.

Finally, in the future, we evaluate the performance of both humans and automatic deepfake text detectors on Authorship Obfuscation techniques.

1.2 Linguistic Patterns of Human vs. Deepfake Texts – RQ1

Linguistics is defined as the study of language and its structural patterns - morphological, syntactical, semantic, etc. Since the capability to generate long-coherent deepfake texts, researchers have attempted to understand the subtle linguistic patterns that distinguish deepfake texts from humans. Thus, in Chapter 3, a Linguistic model is proposed that combines psycholinguistics features (LIWC), Entropy (average number of unique characters), and readability scores (grade level of the author). Using these features with a classical machine-learning model - Random Forest, an interpretable deepfake text detector is built. This model significantly outperforms other baseline models, achieving a 90% F1 score. Some of the interesting findings are: (1) in terms of entropy there is no significant difference between human-written and deepfake texts. (2) the more sophisticated deepfake text-generators are harder to detect because just like humans they use writing articles (a, an, the) well. (3) humans and other human-like deepfake text generators are more evasive in writing.

1.3 Deepfake Text Detector – RQ2

In Chapter 2, we survey the Authorship Attribution models that researchers have proposed for deepfake text detection. This discusses in detail the taxonomy for deepfake text detection models. See Figure 2.5 for this taxonomy. Automatic deeepfake text detectors fall into one of the 4 categories - Stylometric, Deep learning-based, Statistical-based, and Hybrid attribution. Hybrid attribution is an ensemble of at least 2 of the other categories. Additionally, Deep learning-based attribution is further divided into 3 sub-categories - Glove-based, Energy-based, and Transformer-based. This dissertation has contributed to 3 out of the 4 deepfake text detectors in categories. The first is a Stylometric deepfake text detector in Chapter 3 which addressed RQ1 with its interpretive qualities. Second, Transformer-based deepfake text detector in Chapter 4, and third, Hybrid deepfake text detector in Chapter 5. The Hybrid deepfake text detector is an ensemble of Transformer-based models and Topological Data Analysis (TDA) techniques. To be specific a Topological layer is added to a RoBERTa-base model, such that RoBERTa weights and topological features are concatenated and used as input for classification.

1.4 Human Evaluation of Deepfake Texts – RQ3

Several researchers have found that humans perform at about random-guessing (i.e., coin toss) level when distinguishing human-written texts from deepfake texts. This is alarming as we are currently in the *age of deepfake texts* due to the ubiquity of deepfake text generators, such as ChatGPT. However, the capability of these generators, being able to generate coherent documents increases their potential to be used maliciously. Such malicious uses include misinformation generation, as well as toxic and harmful content generation. Therefore, it is imperative that not only automatic deepfake text detectors are accurate, but humans are also able to accurately distinguish human-written texts from deepfake texts. Thus, in chapter 4, we evaluate humans' performance on deepfake text detection by recruiting participants from Amazon Mechanical Turk (AMT). We run 2 exploratory studies - (1) given one article, vote if it is machine-generated or not; (2) given two articles (human-written & machine-generated), vote which of the two is machine-generated. The results suggest that there is a lot of room for improvement.

Therefore, we propose a more thorough study as well as a technique to improve human performance on deepfake text detection in Chapter 6. For this comprehensive study, we investigate the performance of the following groups on the task of deepfake text detection: (1) English experts vs. English non-experts; (2) Individual vs. Collaboration; (3) Asynchronous vs. Synchronous collaboration.

Chapter 2 | Related Work

2.1 Introduction

Natural Language Generation (**NLG**) is a broad term for AI techniques to produce high-quality human-understandable texts in some human languages, and often encompasses terms such as machine translation, dialogue generation, text summarization, data-to-text generation, Question-Answer generation, and open-ended or story generation [5, 6]. Among these, in particular, this survey focuses on the open-ended text generation aspect of NLG. Since the advent of the Transformers architecture in 2018, the field of NLG has experienced exponential improvement. Before 2018, leading NLG models were only able to generate a few sentences coherently. However, after adopting the Transformer architecture into deep learning-based Language models (LMs), NLG models could generate more than a few sentences (i.e., $\geq$ 200 words) coherently. GPT-1 [7] by OpenAI is one of the first such NLG models. Since then, many other Transformer-based LMs with the capacity to generate long coherent texts have been released (e.g., FAIR [8, 9], CTRL [10], PPLM [11], T5 [12], WuDao [1]). In fact, as of February 2023, huggingface's [13] model repo houses about 8,300 variants of text-generative LMs[2]. In this survey, we refer to these LMs as **Deepfake Text Generator** (**DTG**) since they are neural network-based LMs with text-generative abilities. Further, we refer to the texts generated by DTG as **"deepfake" texts** [3], as opposed to normal texts written by humans as **human texts**.

[1]https://github.com/BAAI-WuDao

[2]`https://huggingface.co/models?pipeline_tag=text-generation`

[3]Other names for deepfake text include *AI(-generated) text* [14], *Machine(-generated/written) text* [1, 2, 15–23], *Artificial text* [24], *Computer-generated text* [25], *Neural text* [1, 26, 27], *Auto-generated text* [28], and *Synthetic text* [29–32].

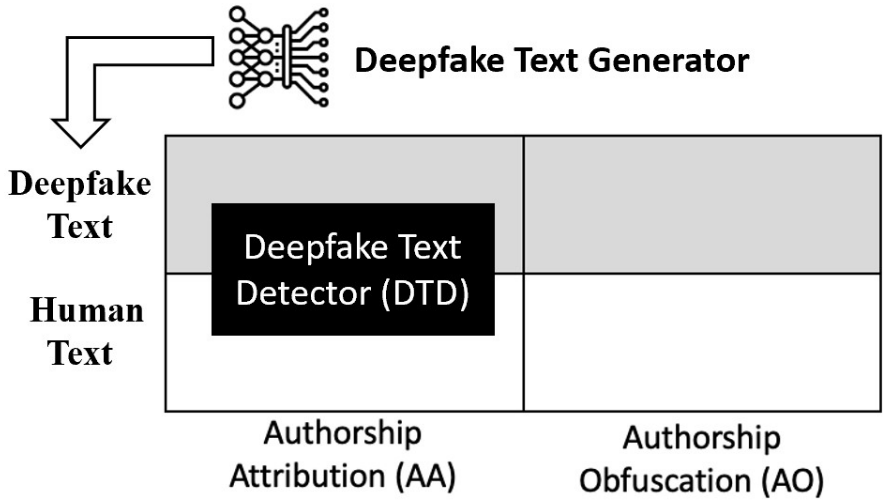

Deepfake Text Generator
Deepfake Text
Human Text
Deepfake Text Detector (DTD)
Authorship Attribution (AA)
Authorship Obfuscation (AO)

Figure 2.1: The figure illustrates the quadrant of research problems where (1) the **GRAY** quadrants are the focus of this survey, and (2) The **BLACK** box indicates the specialized binary AA problem to distinguish deepfake texts from human texts.

As the qualities of DTGs improve, deepfake texts become more easily misconstrued as human-written [1, 2, 16, 33, 34], exacerbating the difficulty of distinguishing deepfake texts from human texts. For instance, therefore, such a text generation capability can be misused to generate misinformation [16, 31, 35], fake reviews [36] and political propaganda [37] at scale with little cost. These problems lead to the need to effectively distinguish deepfake texts from human texts, the so-called **Deepfake Text Detection** (**DTD**) problem, which is a sub-problem of a widely studied problem in the privacy community–i.e., authorship attribution. In fact, two interlocking research questions in privacy research, heavily studied but of growing interest, are **Authorship Attribution** (**AA**) and **Authorship Obfuscation** (**AO**). Given an artifact, especially a text t in question, an AA solution aims to accurately attribute t to its true author out of k candidate authors while an AO solution aims to modify t to hide its true authorship. Therefore, DTD is a specialized case, a Turing Test, of AA with $k = 2$ authors (i.e., human vs. machine). Figure 2.1 illustrates the quadrant of research problems, while Figure 2.2 illustrates how both AA and AO problems work hand-in-hand.

Traditionally, the notion of authorship and its accompanying privacy concern in both AA and AO problems are only toward *human* authors. However, with the arrival of generative AI technologies and due to the potential threats of misused deepfake texts, one now has to consider authorships by humans, machines, or their combination, and re-

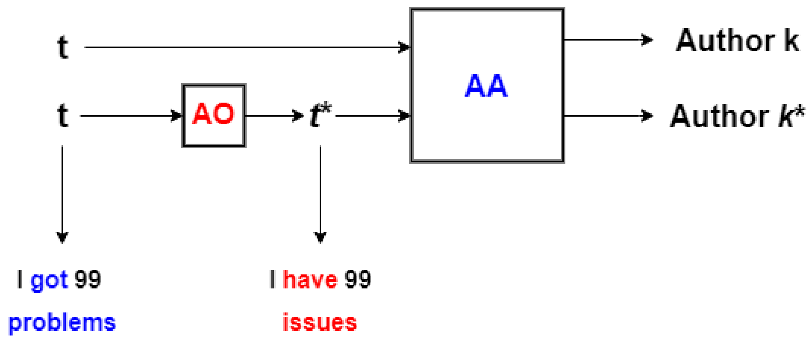


Figure 2.2: Illustration of both AA and AO problems on deepfake texts

thinks about effective solutions for both AA and AO problems for deepfake texts. Hence, to guide these developments, in this survey, we provide a detailed analysis of both AA and AO problems, their existing solutions, and our perspective on the open challenges. As both AA and AO problems are essentially computational learning problems, we discuss the landscape from *A Data Mining Perspective* and call attention to the security challenges that need to be solved. We believe that the issues of these novel AA/AO problems for deepfake texts are "nuanced" and therefore require nuanced solutions from the Data Mining and Machine Learning community.

2.2 Deepfake Text Generation

We first select a handful of Deepfake Text Generator (DTG) that we focus on in this survey, and introduce a list of popular datasets with deepfake texts.

2.2.1 Deepfake Text Generators (DTGs)

Those DTGs studied in this survey are large-scale probabilistic LMs that are capable of generating long-coherent texts (e.g., ≥ 200 words). These LMs are trained on massive amounts of unstructured texts. Based on its architecture structure (e.g., encoder-decoder or decoder only), these LMs use a prompt, a snippet of human-written text, to guide the generation of texts, emulating the most similar style from the training set and predicting one token at a time. Recent works such as [5, 6] survey these DTGs in detail. The progress of DTGs in recent years has been expeditious. As shown in Figure 2.3, for instance, the sizes of DTGs with respect to their parameters are growing at an exponential rate, yielding rapid improvement in the quality of deepfake texts, thus exacerbating the

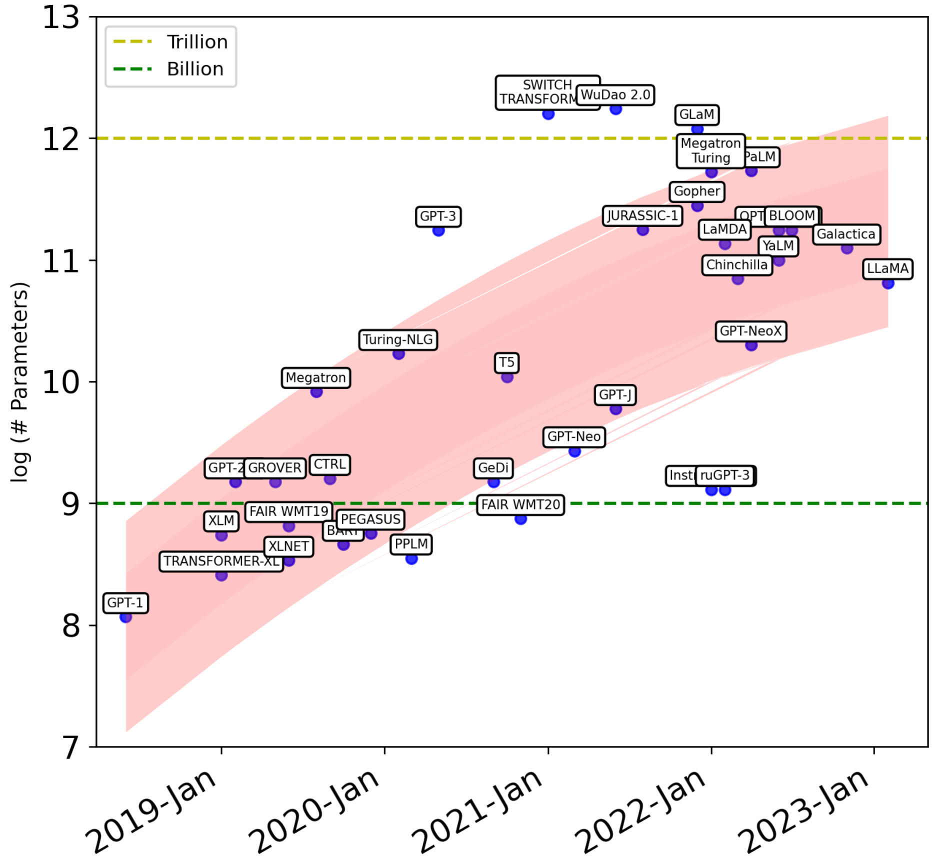


Figure 2.3: Evolution of Deepfake Text Generators (DTGs) from 2018 to 2023 (Y-axis is a log plot of # of parameters).

AA/AO problems. Table 2.1 describes a summary of state-of-the-art DTGs, where many entries are drawn from [1, 15].

Within LMs, in particular, hyperparameters matter a great deal when generating texts. The choice of these hyperparameters, referred to as *decoding strategies*, greatly affects the quality of generated deepfake texts. According to [52], there are 6 *decoding strategies*: (1) *Greedy sampling* selects the best probable word, (2) *Random sampling* does a stochastic search for a sufficient word, (3) *Top-k sampling* samples from top-k most probable words, (4) *Beam search* searches for most probable candidate sequences, (5) *Nucleus (Top-p) sampling* samples similar to top-k, but its focus is on the smallest possible set of top words, such that the sum of their probabilities is $\geq$ p, and (6) *Temperature* scales logits to either increase or decrease the entropy of sampling. DTG models often use *Top-k sampling*, *Beam search*, *Nucleus (Top-p) sampling*, and *Temperature* decoding

DTG	Author	Description
GPT-1 [7]	OpenAI	It used Transformers to model a simple concept - to predict the next token, given the previous token.
GPT-2 [38]	OpenAI	GPT-1 scaled up. There are 4 GPT-2 pre-trained models - *small* (124 million parameters), *medium* (355 million parameters), *large* (774 million parameters), and *x-large* (1558 million parameters)
GPT-3 [31]	OpenAI	GPT-2, scaled up - increasing parameter and train data size.
GROVER [16]	AllenAI	Similar to GPT-2 architecture and trained to generate political news. There are 3 pre-trained models: *GROVER-base*, *GROVER-large*, *GROVER-mega*
CTRL [10]	Salesforce	Conditional Transformer LM For controllable generation uses control codes to guide generation
XLM [39]	Facebook	A Cross-lingual Language Model trained on various languages. Only the English model is used for AA
XLNET [40]	Google	A generalized auto-regressive pre-training method that adopts the Transformer-XL framework
FAIR_wmt [8, 9]	Facebook	FAIR_wmt has 3 language models - English, Russian, and German. Only the English model[4] is used, which has 2 models - *WMT19* [8] and *WMT20* [9].
TRANSFORMER_XL [41]	Google	Another Transformer model that learns long-term dependency to improve long coherent text generation
PPLM [11]	Uber	The Plug and Play Language Models (PPLM) model upon GPT-2 by fusing the GPT-2 medium with a bag of words (BoW) models. These BoW models are *legal, military, monsters, politics, positive_words, religion, science, space, technology*. PPLM can plug in any GPT-2 pre-trained model to generate texts
Switch Transformer [42]	Google	Google uses a switch Transformer to build a sparse neural LM with 1.6T parameters are built
GPT-Neo [43]	EleutherAI	EleutherAI replicates GPT-3's architecture. There a 2 model sizes - 1.3B and 2.7B parameters
GPT-NeoX [44]	EleutherAI	A 20 billion parameter autoregressive replication of GPT-3.
GPT-J [45]	EleutherAI	A 6B parameter model similar to the GPT-Neo and GPT-NeoX that uses Mesh Transformer JAX [46] framework to train the model with Pile[5] dataset, a large curated dataset created by EleutherAI
T5 [12]	Google	An encoder-decoder text-to-text Transformer-based model. T5 has 5 pre-trained models - *T5-small, T5-base, T5-large, T5-3b,* and *T5-11b*
BART [47]	Facebook	This is another encoder-decoder Transformer-based LM, most effective when fine-tuned
PaLM [48]	Google	PaLM stands for Pathways Language Model. It is a dense decoder-only Transformer-based model trained with [49]'s pathways system framework
OPT-175B [50]	Meta	Meta's response to GPT-3. OPT-175B uses a similar framework to GPT-3 but the training costs 1/7th the carbon footprint of GPT-3.
GeDi [51]	Salesforce	GeDi stands for Generative Discriminator Guided Sequence Generation. Similar to PPLM, GeDi controls text generation using small LMs as generative discriminators

Table 2.1: Description of state-of-the-art Deepfake Text Generators (DTGs).

strategies as they produce higher quality texts than other decoding strategies. In fact, [52] reports that deepfake texts generated with the *Nucleus (Top-p) sampling* strategy are more challenging to attribute authorships.

2.2.2 Deepfake Text Datasets

To investigate both AA/AO problems for deepfake texts, one needs benchmark datasets of deepfake texts. Table 2.2 describes a list of publicly available datasets that contain deepfake texts. Labels of the texts in the datasets are either binary (neural vs. human text) or multi-class (having multiple deepfake texts generated by different DTGs vs. 1 human label). The majority of recent studies have focused on the binary case of the Turing Test to check if a given text is written by a human author or machine (i.e., one of DTGs). Researchers utilized clever labeling and generation methods to build datasets: (1) *Binary dataset*: Researchers first collect human-written texts (e.g., news, blogs, stories, or recipes) and use snippets of these texts as prompts to the chosen DTG to generate a machine-written (neural) text. (2) *Multi-class dataset*: Starting with human-written texts as prompts, this generates multiple deepfake texts by using different DTG architectures (i.e., human vs. GPT-1, GROVER, PPLM, etc.), different pre-trained sizes of the same

Name	**Description**	**Category**	**Domain**	**Labels**
GPT-2 dataset [7]	250K Webtext (Human dataset) vs. 250K GPT-2 (small, medium, large, & XL)	Binary	News	GPT-2 & Human
GROVER dataset [16]	Using April 2019 news articles as the prompt, GROVER-Mega generated news articles	Binary	News	GROVER & Human
TuringBench-AA [1]	Used 10K human-written news articles (mostly Politics) from CNN, etc. to generate 10K articles each from 19 DTGs.	Multi-class	News	Human & 19 Machine labels (GPT-1, GPT-2 small, etc.)
TuringBench-TT [1]	The same dataset as *TuringBench-AA*, except that the datasets are 19 versions of human vs. each of the 19 DTGs.	Binary	News	19 Human vs. Machine combinations (GPT-2, etc.)
Authorship Attribution-AA [15]	Used 1K human-written articles to generate 1K articles each from 8 Artificial Text Generators	Multi-class	News	1 human vs 8 Machine labels (GPT-1, GPT-2, etc.)
Authorship Attribution-TT [15]	The same dataset as *Authorship Attribution-AA* except that the datasets are 8 versions of human vs. each of the 8 DTGs	Binary	News	8 humans vs Neural combinations
Authorship Attribution-TT mix [15]	The same dataset as *Authorship Attribution-AA* except that the dataset is human vs. Machine (which is a mixture of all the 8 DTGs)	Binary	News	1 human vs Machine (8 different machines)
Academic Papers & Abstracts [60]	2 datasets - (1) FULL: using a short prompt for a human-written paper generated an academic paper using GPT-2; (2) PARTIAL: Replacing sentences from an Abstract with Arxiv-NLP model generations	Binary	Academic Abstracts	Human & Machine
Hybrid Human-Machine Text [61]	Using human-written text in domains - News, Reddit, and Recipes to generate continuations of the text using GPT-2 XL	Binary	News, Reddit, Recipes	Human prompt & Machine texts
Amazon Reviews [36]	Fine-tuned GPT-2 on 3.6 M Amazon and 560K Yelp reviews	Binary	Reviews	Human & Machine
Human-Machine Pairs [57]	Generated texts with GROVER mega and GPT-2 XL with top-p decoding strategy. Paired human-written texts with a similar neurally generated version	Binary (Human-Machine pairs)	Online forums & News	Human & Machine
NeuralNews dataset [23]	Using the GoodNews [62] dataset as the human-written prompt to generate texts with GROVER. Real images are included in each of the articles.	Binary	News	Human & Machine
SynSciPass [63]	Built dataset using 3 potential sources of deepfake text: (1) open-ended text generators like GPT-2 & BLOOM (2) paraphrase models like SCIgen and PEGASUS and (3) translation models like Spinbot, real, Google translate, and Opus.	Multi-class	scientific articles	generate, translate, paraphrase, & human
TweepFake [64]	Collected tweets generated by Twitter bots and grouped them into tweets generated by GPT-2, RNN, and other bots	Binary	Tweets	Human & Machine

Table 2.2: Datasets with Deepfake Texts

DTG architecture (i.e., human vs. GPT-2 small vs. GPT-2 medium vs. GPT-2 large vs. GPT-2 XL, etc.), different decoding strategies (i.e., human vs. GPT-2 top-k vs. GPT-2 top-p, etc.).

Model Name	Classifier Type	Category	Learning Type	Interpretable	Training dataset
GROVER detector [16]	DL (Transformer-based)	Binary	Supervised		GROVER
GLTR [2]	Statistical	Binary	Unsupervised	✓	GPT-2
GPT-2 detector [65]	DL (Transformer-based)	Binary	Supervised		GPT-2
OpenAI detector [65]	DL (Transformer-based)	Multi-class	Supervised		GPT-2 & TuringBench-AA
RoBERTa-TT [1]	DL (Transformer-based)	Binary	Supervised		TuringBench-TT
BERT-TT [1]	DL (Transformer-based)	Binary	Supervised		TuringBench-TT
RoBERTa-Multi [1]	DL (Transformer-based)	Multi-class	Supervised		TuringBench-AA
BERT-Multi [1]	DL (Transformer-based)	Multi-class	Supervised		TuringBench-AA
TDA-based detector [24]	Hybrid	Binary	Supervised	✓	GPT-2
FAST [27]	Hybrid	Binary	Supervised		GPT-2 & GROVER
Energy discriminator [17]	DL (Energy-based)	Binary	Supervised		GROVER
MAUVE [20]	Statistical	Binary	Unsupervised	✓	GPT-2 & GROVER
Distribution detector [21]	Statistical	Binary	Unsupervised	✓	GPT-2
Feature-based detector [19]	Stylometric	Binary	Supervised	✓	GPT-2, GPT-3, & GROVER
Linguistic model [15]	Stylometric	Multi-class	Supervised	✓	Authorship Attribution-AA
DIDAN [23]	Hybrid	Binary	Supervised		Fake images w/ Human vs. GROVER news
XLNet-FT [30]	DL (Transformer-based)	Multi-class	Supervised		GPT-1, GPT-2, XLNet, & BART Reddit posts
Constra-DeBERTa [58]	DL (Transformer-based)	Multi-class	Supervised		TuringBench
Fingerprint detector [29]	Hybrid	Multi-class	Supervised		GPT-2 bot subreddit posts
DistilBERT-Academia [60]	DL (Transformer-based)	Binary	Supervised		GPT-2 Academia abstract & paper
Sentiment modeling detector [36]	DL (Glove-based)	Binary	Supervised		GPT-2 Amazon Reviews
BERT-Defense [33]	DL (Transformer-based)	Binary	Supervised		GPT-2 Large WebText
RoBERTa-Defense [55]	DL (Transformer-based)	Binary	Supervised		GROVER
RoBERTa w/ GCN [66]	DL (Transformer-based)	Binary	Supervised		GPT-2
DeBERTa v3 [63]	DL (Transformer-based)	Multi-class	Supervised		GPT-2, BLOOM, PEGASUS, OPUS, SCIgen, Spinbot
Ensemble [67]	DL (Transformer-based)	Binary	Supervised		TweepFake
CoCo [68]	Hybrid	Binary	Supervised	✓	GPT-2 & GROVER
DetectGPT [69]	Statistical	Binary	Unsupervised	✓	GPT-2, OPT-2.7, GPT-Neo-2.7, GPT-J, & GPT-NeoX

Table 2.3: Authorship Attribution models (Binary & Multi-class) for Deepfake Text Detection

2.3 Authorship Attribution for Deepfake Texts

Traditional AA problem studies the attribution of an author to a piece of written text out of a number of possible authors. However, in the literature, researchers have also studied a few variations of the AA problem. For instance, the *Author Verification (AV)* problem [70, 70–74] studies if the given two texts, t_1 and t_2, are written by the same author? With the rise of deepfake texts, in addition, a specialized case of AA problem, DTD [15–17, 23, 65, 75], studies: By and large, however, a good solution for the standard AA problem can lead to a good solution for other variations of the AA problem. As such, we focus on the survey of the standard AA problem for deepfake texts. Thus, our paper formally defines the AA task as follows. See Table 2.3 for a detailed taxonomy of AA solutions for deepfake text detection and Figure 2.4 for the count of AA solutions in each category.

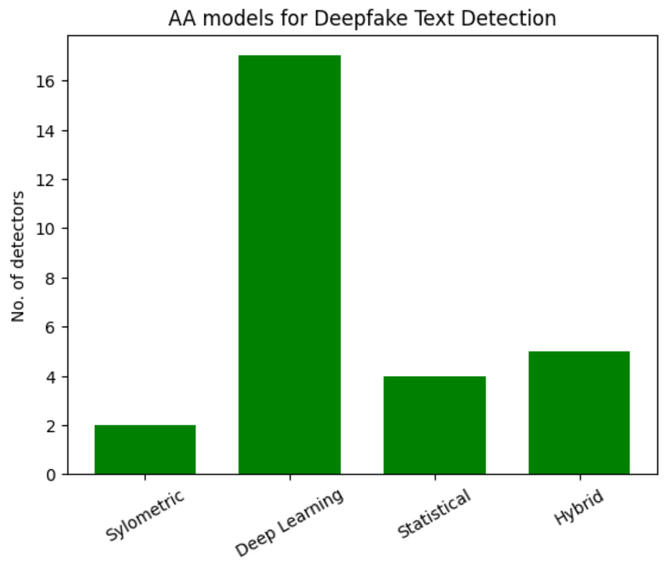


Figure 2.4: Number of AA solutions for Deepfake Text Detection per category in the Taxonomy. Y-axis is the number of detectors.

Definition of AA for Deepfake Texts. *Given a text t, the AA model $F(x)$ attributes the text t to its true author k–i.e., k=$F(x)$, which can be either a human or a DTG author.*

In the following, we survey recent AA solutions that are capable of handling deepfake texts in different ways, as illustrated in Figure 2.5: *Stylometric Attribution*, *Deep Learning-based Attribution*, *Statistical Attribution*, and *Hybrid Attribution*.

2.3.1 Stylometric Attribution

Stylometry is the statistical analysis of the style of written texts. In traditional AA, stylometric classifiers are built using classical machine learning models trained on ensembles of style-based features such as N-grams, Part-of-Speech (POS), WritePrints [22], LIWC (Linguistic Inquiry & Word Count) [76], Readability score, and Empath [77]. This has been shown to be a successful approach for traditional AA tasks [78]. Due to such success, these models have been adopted and customized to the task of DTD.

The first attempt at a stylometric classifier to solve the AA task for $k > 2$ authors is the *Linguistic model* proposed by [15]. It trains a Random Forest classifier with the Authorship Attribution-AA dataset in Table 2.2 and extracts an ensemble of stylometric

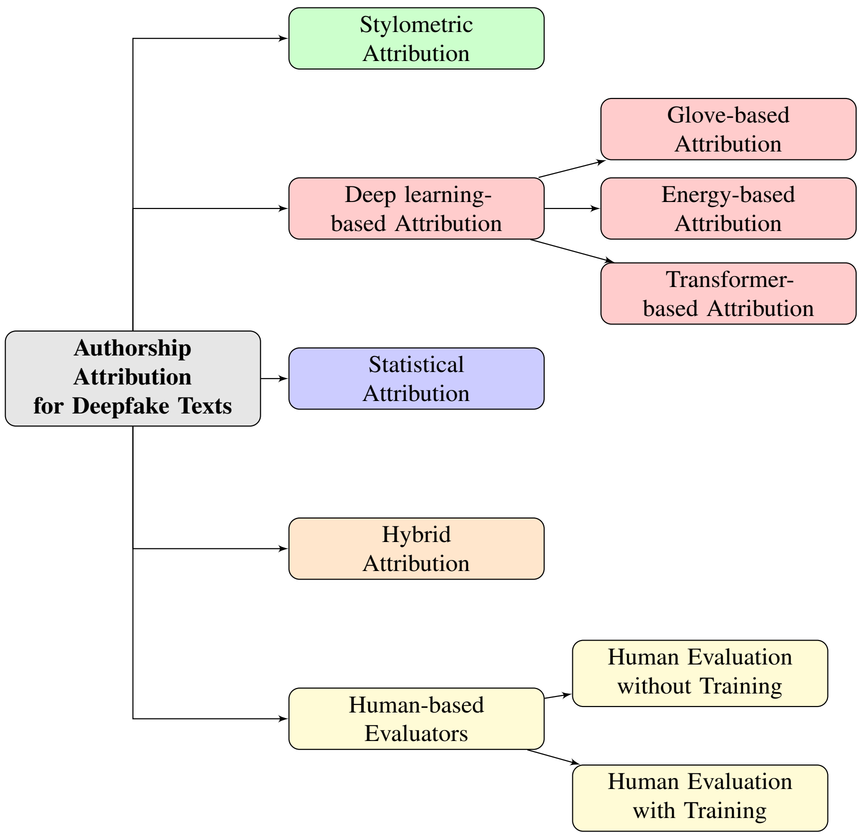

Stylometric Attribution
Glove-based Attribution
Deep learning-based Attribution
Energy-based Attribution
Transformer-based Attribution
Authorship Attribution for Deepfake Texts
Statistical Attribution
Hybrid Attribution
Human Evaluation without Training
Human-based Evaluators
Human Evaluation with Training

Figure 2.5: Taxonomy of Authorship Attribution models for Deepfake Text Detection

features (e.g., *entropy*, *readability score*, & *LIWC* (Linguistic Inquiry & Word Count) [76]). The entropy feature counts the number of unique characters in the text. Readability scores represent the estimated educational level of the author of a piece of text based on lexicon usage. LIWC is a psycho-linguistic dictionary that counts the frequency of words that represents a psychological emotion or linguistic structure [76]. This *Linguistic model* achieves a 90% F1 score and outperforms all the other deep learning-based models. However, this superior performance is a result of the small size of the dataset (only about 1k per data label) [15]. Scaling up the data size in terms of labels and examples will make the AA task harder, and therefore cause the *Linguistic model* to underperform. This claim is confirmed in their second work using the TuringBench dataset [1]. They compared SOTA deep-learning-based models (BERT and RoBERTa) with several stylometric classifiers - SVM (3-grams), WriteprintsRFC, Random Forest (w/ TF-IDF), Syntax-CNN, Ngram CNN, and N-gram LSTM-LSTM. RoBERTa outperforms all the stylometric classifiers with about a 10-22% increase in F1 scores.

To further explore the benefits of stylometric features leveraged in the traditional AA community, [19] proposes a clever way to use them. This solution aims to solve the special case of AA, *Turing Test* (TT). First, they identify different issues with DTGs which can be captured by specific types of stylometric features. These issues in deepfake texts are categorized into 4 types: (1) *Lack of syntactic and lexical diversity* which can be captured with Named Entity-tags, POS-tags, and neuralcoref extension[6] (a tool for using a neural network to annotate and resolve coreference clusters) to detect coreference clusters; (2) *Repetitiveness of words* which can be captured by collecting the number of stop-words, unique words, and words from "top-lists" of total words in a text. Also, a "conjunction overlap" measure is defined to calculate the overlap of the top-k words ($k = 100, 1K, 10K$); (3) *Lack of coherence* which can be captured using entity-grid representation to track the appearance of the grammatical role of entities. They also use neuralcoref to detect coreference entity clusters; (4) *Lack of purpose* which is captured using a lexicon-package, empath [77] containing 200 linguistic features [19]. To evaluate the generalizability of these features, an ensemble of all the features is used to build a classifier - *Feature-based detector*. This detector is trained and tested on different sizes of the GPT-2 models. It is further evaluated on GPT-3 and GROVER texts. The classifier performs consistently in detecting texts generated by GPT-3, GROVER, and different model sizes of GPT-2, suggesting it is generalizable to different DTG model sizes [19]. Further results suggest that some of the 4 categories of issues are prevalent in the top-k decoding strategy. Also, more quality-focused features (especially ones focused on Lexical diversity) perform better than statistical features such as the TF-IDF baseline.

Scaling up and creating a more realistic scenario, [29] collect 108 SubReddit blog posts generated by GPT-2 fine-tuned on 500K subreddit posts and comments. Every 108 labels indicate the 108 users of SubReddit (r/SubSimulatorGPT2). These 108 authors are detected using a set of features called "writeprints" features for the AA model [29]. Writeprints [22] is a stylometric feature that collects lexical, content-based, and idiosyncratic features as the baseline. The writeprints classifier underperforms, compared to the RoBERTa-based baseline models. Similarly, a stylometric classifier with 791 stylometric features based on [79] is used to detect deepfake texts. This classifier has 4 categories of features: *Character, word, sentence,* and *Lexical Diversity* features. The classifier is an ensemble of classical ML models such as Random Forest and SVM and the stylometric features. BERT, a non-stylometric classifier, outperforms these stylometric

[6]https://github.com/huggingface/neuralcoref

classifiers significantly [80].

Finally, stylometric classifiers are best used when the dataset size is small. When data size increases, these models underperform, allowing deep learning-based models to outperform significantly. Thus, we conclude that stylometric classifiers can only be considered good baselines since they underperform when the problem scales up. Another limitation of stylometry is that it fails to detect neural misinformation due to DTG's capacity to generate consistent misinformation [18].

2.3.2 Deep Learning-based Attribution

Stylometric classifiers struggle to accurately assign the true authorship to human vs. deepfake texts. In Section 2.3.1, we observe that some of the stylometric classifiers were outperformed by deep learning-based models. Additionally, [18]'s findings of stylometry failing to detect neural misinformation, further calls for a different technique to solve the AA task for DTD. Therefore, researchers have adopted and advanced deep learning-based techniques for the attributing of neural vs. human text. These models can be further categorized into 3 types of deep learning-based classifiers - *Glove-based, Energy-based,* and *Transformer-based* Attribution.

2.3.2.1 Glove-based Attribution

Glove is an unsupervised learning algorithm for extracting the representation of words. It aggregates global word-word
co-occurrence statistics from a piece of text [81]. Using GloVe word embeddings with RNN and LSTM-based neural networks was considered SOTA until, 2018 (birth of BERT [82]). Thus Glove-based classifiers now provide a good baseline for text classification tasks. Some of the best-performing AA classifiers in the traditional AA communities are an ensemble of stylometric features + GloVe pre-trained models w/ a neural network architecture. Several Glove-based classifiers have been used as baselines for the DTD task. *Syntax-CNN* [83], *N-gram CNN* [84], and *N-gram LSTM-LSTM* [85] are baselines for [1]. *Embedding*, *RNN*, *Stacked-CNN*, *Parallel-CNN*, and *CNN-RNN*, are baselines for [15]. They demonstrated that Glove-based classifiers are unsuitable for solving the AA problem when there are $k > 2$ authors. Furthermore, the *Fingerprint detector* is compared with 4 baseline models - *Gaussian Naive Bayes*, *Random Forest*, *Multi-layer Perceptron*, and *CNN* classifiers using the Glove word embeddings of the

data. They underperform significantly [29].

Lastly, *Sentiment modeling detector*, a variation of sentiment neuron used to learn a single-layer multiplicative LSTM (mLSTM) [86] is used to detect texts generated by GPT-2 [36]. The goal is to force the model to focus on a specified sentiment [36]. This model outperforms and sometimes performs comparably with the baseline - original mLSTM model [87], achieving a 70% accuracy in detecting GPT-2 generated Amazon and Yelp reviews [36].

2.3.2.2 Energy-based Attribution

Energy-based models (EBMs) are un-normalized generative models based on energy functions [88]. Using the energy functions, EBMs model the probability distribution of its training data and generates high quality data similar to the training set [88]. It is also able to adapt to changes in the Language model. Due to this capability, *Energy-based classifier* is proposed by [17] to detect deepfake texts. This classifier is trained on 3 datasets of different domains - *Books*, *CCNews*, and *Wikitext*. Three sizes of the GPT-2 model are used for the generator architectures and three architectures are used for the energy function - *Linear*, *BiLSTM*, and *Transformer*. Their findings suggest that: (1) as the DTG increases in size, the harder the AA task becomes; (2) the biggest energy function (i.e.*Transformer*) performs the best in detecting texts generated from large language models (e.g., GPT-2 Large & XL); (3) as the length of texts increases, the task becomes even more non-trivial; and (4) the classifiers are able to generalize to data that it is not trained on.

In addition, EBMs are very expensive to train and do not scale well [17]. While the *Energy-based classifier* performs well in the AA problem, achieving over a 90% in all experiments, applying the classifier to a much larger dataset is too expensive to justify.

2.3.2.3 Transformer-based Attribution

Since the advent of the Transformer architecture, the current SOTA text classification models are Transformer-based models. Based on Section 2.3.2.2, we observe that large models are better at detecting deepfake texts. However, since EBMs are too expensive, several researchers have adopted Transformer-based models (i.e., BERT, RoBERTa, etc.) for the AA tasks. In Section 2.3.1, most of the stylometric classifiers were outperformed by Transformer-based classifiers. This further supports the application of this classification technique to the AA problem.

GROVER detector[7] [16] is trained on texts generated by the GROVER DTG. It is built with similar architecture as the GPT-2 classifier. *GROVER detector* has been evaluated on deepfake texts generated by different DTGs (GPT-2, FAIR, PPLM, etc.) [1, 15, 27, 89]. It performs well at detecting deepfake texts generated by older DTGs (2018-2019), however, struggles at detecting more recent DTGs accurately. For instance, *GROVER detector* achieved a 58% F1 score in detecting GPT-3 texts with the TuringBench dataset [1]. Next, GPT-2 has a detector trained to detect texts generated by GPT-2 - *GPT-2 detector*[8] [65]. Just like *GROVER detector*, *GPT-2 detector* has also been evaluated on deepfake texts generated by different DTGs [1, 15, 53, 90] and more easily detects older DTGs than newer DTGs. This is confirmed with *GPT-2 detector*'s performance in detecting GPT-3 texts, achieving a 53% F1 score [1]. The reason is that newer DTGs, such as GPT-3 tend to generate more human-like texts which confuse SOTA older AA models, like *GPT-2 detector*.

There are two RoBERTa-based models (base & large) trained on GPT-2 dataset[9] in huggingface repo[10],[11]. We call both the base and large models, *OpenAI detector*. This AA model has been evaluated on deepfake texts generated by different DTGs [1, 53]. *OpenAI detector* is the same model as *GPT-2 detector*, except that *OpenAI detector* has been re-purposed for the AA multi-class setting, while *GPT-2 detector* remains for the AA binary setting. *OpenAI detector* performs comparably to the AA models - *BERT-Multi* and *RoBERTa-Multi* when evaluated on the TuringBench-AA dataset [1]. *BERT-Multi* and *RoBERTa-Multi* are BERT and RoBERTa base models, respectively trained on the TuringBench-AA dataset. *BERT-TT* and *RoBERTa-TT* outperform *GROVER detector* and *GPT-2 detector* when evaluated on the TuringBench-TT dataset [1]. *BERT-TT* and *RoBERTa-TT* are BERT and RoBERTa base models, respectively trained on the TuringBench-TT dataset. *BERT-TT*, outperforms all the models, including *RoBERTa-TT* significantly. Furthermore, for all 19 pairs of human vs. DTG, no model consistently outperforms all other models. In fact, *GROVER detector* and *GPT-2 detector* performs poorly in detecting texts generated by GROVER and GPT-2, respectively [1].

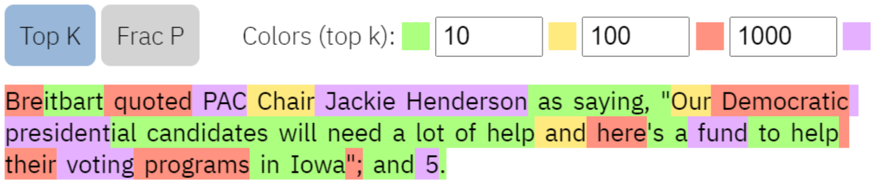
XLNet-FT is a fine-tuned XLNet classification model trained to detect texts generated

[7]https://grover.allenai.org/detect
[8]https://huggingface.co/openai-detector/
[9]https://github.com/openai/gpt-2-output-dataset
[10]https://huggingface.co/roberta-base-openai-detector
[11]https://huggingface.co/roberta-large-openai-detector

by GPT-2 [30]. The generalizability of the model is evaluated on different subreddit post domains. *XLNet-FT* performs consistently, achieving over a 90% accuracy in all experiments, suggesting that it is generalizable [30]. However, when *XLNet-FT* is further evaluated on deepfake texts generated by top-p and top-k decoding strategy, there is a significant drop in accuracy, suggesting that the AA model may not be generalizable.

Using an *in-the-wild* dataset concept, *RoBERTa-Defense* is evaluated on 4 types of *in-the-wild* datasets [55]. *In-the-wild* datasets are test sets generated from an entirely different DTG from the training set. *RoBERTa-Defense* is trained on human & GROVER Real news in Table 2.2 and compared to other SOTA AA models - *GLTR* (with two different LMs - BERT & GPT-2 which results in *GLTR-BERT* & *GLTR-GPT2*), *GROVER detector*, *BERT-Defense*, and *FAST*. *RoBERTa-Defense* outperforms all other models significantly. *BERT-Defense*, a BERT model fine-tuned on GPT-2 Large Webtext dataset in Table 2.2 from which *RoBERTa-Defense* is inspired is evaluated with different decoding strategies [33]. *BERT-Defense* is trained and tested on deepfake texts generated by different decoding strategies - top-k, top-p, untruncated random, and mixed (i.e., dataset containing equal amounts of each strategy) [33]. The classifier trained on the mixed dataset is the most generalizable classifier. Similarly, *RoBERTa, BERT*, *ELECTRA* [91], and *ALBERT* [92] are evaluated on *in-the-wild* dataset [93]. These models are evaluated, specifically on out-of-domain COVID-19 human-written vs. neural news. The neural news is generated with GPT-2 small, medium, large, XL, and GPT-Neo using top-p and top-k decoding strategies [93]. *ELECTRA* performs better at generalizing to out-of-domain deepfake texts, achieving an average accuracy of 86% on all out-of-domain datasets.

Due to the nuances of deepfake texts, [58] proposes to combine the advantages of contrastive learning [94] with a Transformer-based classifier. Thus, they propose *Constra-DeBERTa* which is a DeBERTa model [95] trained with a contrastive learning approach. Contrastive learning is a technique that clusters similar examples together and separates dissimilar examples in a representation space [58, 94]. However, while *Constra-DeBERTa* outperforms other SOTA traditional AA models, it only performs comparably to *RoBERTa-Multi* on detecting the TuringBench dataset. Similarly, [63] trains *DeBERTa v3* [96] on the SynSciPass dataset to answer the question, if a piece of text is neurally generated, how was it generated? The answer choices are *generated, paraphrased*, or *translated* vs. human-written. Using this dataset, *DeBERTa v3* achieves a 99.6% F1 score (i.e., F1 score is the harmonic mean of precision and recall which are



Colors (top k): 🟩 10 🟨 100 🟥 1000 🟪

Breitbart quoted PAC Chair Jackie Henderson as saying, "Our Democratic presidential candidates will need a lot of help and here's a fund to help their voting programs in Iowa"; and 5.

Figure 2.6: [1] used GLTR [2] on GPT-3 texts. Green represents the most probable words (top-10); yellow the 2nd most probable (next top-100 probable words); Red the least probable (next top-1000 probable words); and purple the highest improbable words. The hypothesis is that deepfake texts are often populated with mostly Green and yellow words. However, we see that texts generated by GPT-3 are very human-like according to the hypothesis.

popular machine learning evaluation metrics). Next, *DistilBERT-Academia* is trained on human vs. GPT-2 academic abstracts and papers [60] and achieves a 62.5% and 70.2% accuracy on the FULL and PARTIAL Academic datasets, respectively. Furthermore, *RoBERTa w/ GCN* (Graph Convolutional Networks) is used to detect human-written news with entities manipulated and replaced by GPT-2 [66]. The GCN [97] model is used to capture factual knowledge of neural vs. human news articles. RoBERTa outperformed the proposed model - *RoBERTa w/ GCN* on most of the GPT-2 detection tasks [66].

Lastly, *ruBERT*, a Russian BERT model is used to distinguish Russian neural texts from Russian human-written texts as a shared task [98]. This fine-tuned *ruBERT* (Russian BERT) achieves 82.6% accuracy for $k = 2$ authors and 64.5% accuracy for $k > 2$ authors [28]. Finally, using an *Ensemble* classifier (BART, BERTweet, and TwitterRoberta), [67] achieves an 84% accuracy in distinguishing between GPT-2 and human tweets. However, using the same model for GPT-3 generated tweets achieves a 54% accuracy [67]. This suggests that GPT-3 generates more human-like tweets than GPT-2.

2.3.3 Statistical Attribution

We observe that while there are some well-performing stylometric and deep learning-based models, there is still a lot of room for improvement, especially in building generalizable models. The biggest feat is in building classifiers that perform consistently well in detecting deepfake texts generated by top-p and top-k decoding strategies. Thus, statistical models are proposed to combat these limitations. To assess the validity of statistical techniques, a hypothesis using k-order Markov approximations are formulated [99]. This statistical formulation proves the hypothesis that human language is stationary and ergodic as opposed to neural language. The formal hypothesis testing framework is used to establish limits in error exponents between human and deepfake text [99]. This suggests that statistical AA models for deepfake texts could be successful. There are currently only four statistical classifiers that capture the writing style of deepfake texts by modeling their statistical distribution.

The first statistical AA classifier proposed for DTD is *GLTR* [2]. *GLTR* performs 3 tests - (1) probability of the word; (2) the absolute rank of the word; (3) the entropy of the predicted distribution to detect deepfake texts. *GLTR* has a demo[12] that highlights words by distribution and is used to assist humans in detecting deepfake texts. See Figure 2.6 [1] to see how *GLTR* detects texts generated by GPT-3. This classifier improved human performance in detecting deepfake texts from 54% to 72%. However, since 2019 when it was built, more sophisticated DTGs have been built. These newer DTGs have more human-like statistical distribution, making it harder for *GLTR* to distinguish deepfake texts from human texts. *GLTR*, especially underperforms in detecting GPT-3 texts, achieving a 35% F1 score, which is significantly less than a random guess (50%) [1].

MAUVE is another statistical classifier [20]. This AA classifier measures the gap between the distribution of human and deepfake texts. Using KL-divergence, *MAUVE* models two types of errors that highlight the unique distributions in human vs. deepfake texts [20]. Human detection of texts generated by GROVER and GPT-2 correlated strongly with *MAUVE*'s highlight of differences between human and deepfake texts. *Distribution detector*, an unsupervised AA model for calculating the distribution of repeated n-grams in deepfake texts is used to detect deepfake texts [21]. The hypothesis is that DTGs are more repetitive than humans which is also one of the hypotheses of [19]. *Distribution detector* achieves over 90% and 80% accuracy in detecting texts generated by GPT-2 using top-k and top-p decoding strategies, respectively.

Most recently, a zero-shot unsupervised deepfake text detector, *DetectGPT* [69], is proposed. The hypothesis of this statistics-based detector is that deepfake texts tend to lie in areas of negative curvature of the log probability function [69]. Therefore, if a piece of deepfake text is perturbed, the curvature of the log probability will still bear a strong similarity to the unperturbed deepfake texts. Hence, [69] considers an AO technique that slightly modifies the original deepfake text, while preserving semantics. After running several perturbation experiments, a threshold for perturbation discrepancy is defined and used to detect deepfake text. Thus, by measuring the curvature of log probability with

[12]http://gltr.io/dist/index.html

the strict constraint of perturbation discrepancy threshold, *DetectGPT* can detect texts generated by a neural method. Finally, *DetectGPT* detects GPT-3 generated texts with an average of 85% AU-ROC, performing comparably to RoBERTa [68]. Lastly, *DetectGPT* has an online demo[13].

2.3.4 Hybrid Attribution

There are advantages in each of the AA model categories, however, each of them is still unable to accurately attribute neural vs. human texts to their authors consistently. Furthermore, the issue of different decoding strategies, also, make the AA models unable to generalize well [19, 33, 52, 55]. Therefore, a few researchers have proposed hybrid classifiers, which are ensembles of two or more of the AA categories.

TDA-based detector, an ensemble of the Transformer-based and statistical AA techniques is used to solve the DTD task. This classifier involves obtaining the attention matrices of BERT's word representations of texts generated by GPT-2 and GROVER. Next, using these BERT word representations, 3 interpretable TDA-based features are extracted: (1) *Topological Features*: Calculating the first 2 betti numbers (i.e., topological features based on the connectivity of n-dimensional simplicial complexes) of the attention matrices; (2) *Features derived from barcodes*: Calculating characteristics of the barcode plots of the persistent homology of the attention matrix; (3) *Features based on the distance to patterns*: Calculating the distance in features in the attention graph. This feature is used to capture linguistic patterns. Finally, *TDA-based detector* is a logistic regression model, trained on an ensemble of the three TDA features [24]. Comparing this model to pre-trained and fine-tuned BERT models, it performs comparably to BERT models fine-tuned on GPT-2 small Webtext, GPT-2 XL Amazon Reviews, and GROVER News [24]. While more analysis is required to understand why the *TDA-based detector* performs well, this approach has interpretable qualities that should be explored in future work.

Fingerprint detector is another hybrid classifier for DTD. It is an ensemble of fine-tuned RoBERTa embeddings and CNN classifier [29]. *Fingerprint detector* solves the AA task by detecting 108 neural authors. The *Fingerprint detector* achieves a 70% accuracy (top-10). This shows promise in the area of detection of deepfake texts *in-the-wild*, where there are $k > 100$ authors. To continue the quest for generalizable classifiers, *FAST* uses a

[13]https://detectgpt.ericmitchell.ai/

Graph Neural Network (GNN) architecture with RoBERTa to capture the factual structure of neural and human texts [27]. It detects deepfake texts by calculating the RoBERTa word embeddings of the texts and then extracting the graphical representation [27]. Next, it uses a GNN to capture sentence representations that consider coherence [27]. The experiments included detecting texts generated by GROVER and GPT-2. *FAST* outperforms *GROVER detector* and other baselines significantly. Surprisingly, it performs the best at detecting human-deepfake text pairs, achieving over 93% accuracy while unpaired texts achieve over 84% accuracy.

CoCo is a coherence-based contrastive learning model [68]. It is architecturally similar to *FAST* in that it uses a graphical neural network to represent the sentences of human-written vs. deepfake texts. Since human-written texts are more coherent than deepfake texts, they sentences share more entities [68]. The texts are represented as RoBERTa embedding weights which are concatenated with the sentence-level graphical representations of the texts. These concatenated features are input for an LSTM with attention. Lastly, *CoCo* is trained using the sum of the coss-entropy loss and contrastive loss [68] to improve model performance. Thus, it achieves an F1 score of 83% and 94% using the full dataset for GROVER, and GPT-2, respectively [68]. Furthermore, calculating the graphical metrics showed that in terms of the number of vertex and edges, human-written texts have significantly more graphical features than deepfake texts.

Lastly, [23] explore the most realistic scenario of misinformation where malicious users of DTGs, pair neurally generated misinformation with fake/real images to increase the authenticity of the news article. *DIDAN* is a multi-modal DTD evaluated on a multi-modal dataset containing both texts and images [23]. This DTD encodes the texts with BERT encoder and investigates Visual-Semantic representations from images and texts. These features are used to evaluate the semantic consistency between linguistic and visual components in a news article [23]. An *authenticity score* is defined to represent the probability of an article being human-written. It is calculated by extracting the co-occurrences of named entities in the news articles and captions [23]. They build different variations of dataset, some only containing text. Using only the text dataset, *DIDAN* is compared to *GROVER detector* as well as other baseline models, and outperforms all of them [23].

Authorship Obfuscation	**Adversarial Attack**
Has a strict definition of writing style	Has a loose definition of writing style
Requires consistent/uniform change in writing style	Does not require consistent/uniform change in writing style
Requires semantics to be preserved	Does not always require semantics to be preserved

Table 2.4: Differences between Authorship Obfuscation and Adversarial Attack

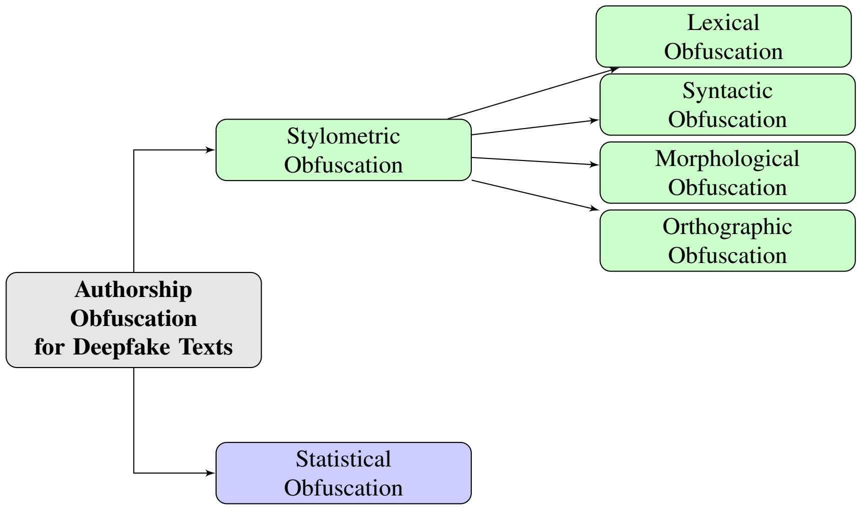


Figure 2.7: Taxonomy of Authorship Obfuscation algorithms/techniques for Deepfake Text Detection

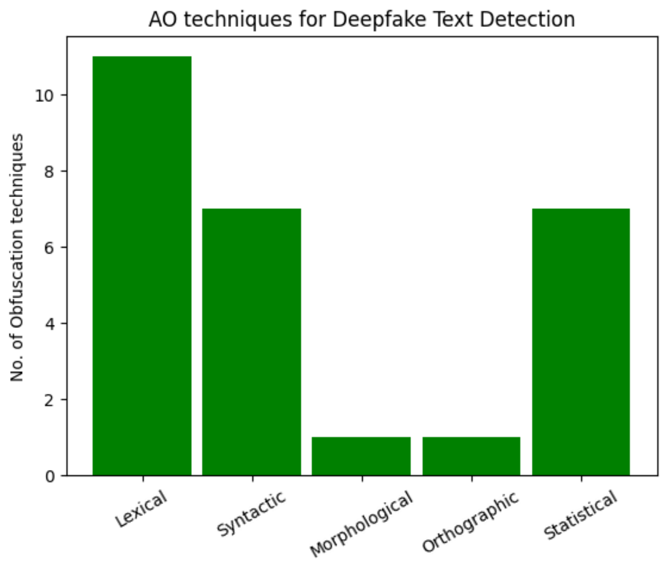


Figure 2.8: Number of AO techniques for Deepfake Text Detection per category in the Taxonomy. Y-axis is the number of detectors.

AO technique	Scenario	Category	Interpretable	Preserves semantics	Obfuscated dataset
Homoglyph [53, 89]	Black-box	Stylometric (Orthographic)	✓	✓	GROVER & GPT-2
Upper/Lower Flip [89]	Black-box	Stylometric (Morphological)	✓	✓	GROVER & GPT-2
DeepWordBug [25, 54]	Black-box	Stylometric (Lexical)			GPT-2 & GPT-3
Misspellings attack [53, 89]	Black-box	Stylometric (Lexical)	✓		GROVER & GPT-2
Whitespace attack [89]	Black-box	Stylometric (Lexical)	✓		GROVER & GPT-2
Deduplicate tokens [57]	Black-box	Stylometric (Lexical)	✓		Human-Machine Pairs
Shuffle tokens [57]	Black-box	Stylometric (Syntactic)	✓		Human-Machine Pairs
Retain only (non)-stopwords [57]	Black-box	Stylometric (Syntactic)	✓		Human-Machine Pairs
Retain tokens in high/low frequency [57]	White-box	Statistical	✓		Human-Machine Pairs
Replace text with likelihood ranks [57]	White-box	Statistical	✓		Human-Machine Pairs
Replace text with specific linguistic features [57]	White-box	Statistical	✓		Human-Machine Pairs
TextFooler [54, 55]	Black-box	Stylometric (Lexical)		✓	GPT-2 medium, GPT-2 XL, GPT-3, GROVER
Varying sentiment [56]	Black-box	Stylometric (Lexical)	✓		GROVER
Source-target exchange [56]	Black-box	Stylometric (Syntactic)	✓		GROVER
Entity replacement [56]	Black-box	Stylometric (Lexical)	✓		GROVER
Alter numerical facts [56]	Black-box	Stylometric (Syntactic)	✓		GROVER
Syntactic perturbations [56]	Black-box	Stylometric (Syntactic)	✓		GROVER
Article shuffling [56]	Black-box	Stylometric (Syntactic)	✓		GROVER
Selecting highest human -class probability [28]	Black-box	Statistical			Russian neural & human-written texts
Adding detector's log-probability to sampling technique [28]	White-box	Statistical			Russian neural & human-written texts
Varying the text decoding strategy (and its parameters) [55]	White-box	Statistical		✓	GPT-2 Large, GPT-2 XL, GPT-3, GROVER
Varying the number of priming tokens [55]	White-box	Statistical			GPT-2 Large, GPT-2 XL, GPT-3, GROVER
DFTFooler [55]	Black-box	Stylometric (Lexical)		✓	GPT-2 Large, GPT-2 XL, GPT-3, GROVER
Random Perturbations [55]	Black-box	Stylometric (Lexical)			GPT-2 Large, GPT-2 XL, GPT-3, GROVER
ALISON [59]	Black-box	Stylometric (Syntactic)	✓	✓	TuringBench
Mutant-X [59]	Black-box	Stylometric (lexical)		✓	TuringBench
Avengers [59]	Black-box	Stylometric (lexical)		✓	TuringBench

Table 2.5: Authorship Obfuscation Techniques for Deepfake Texts. *With cited papers that implemented them.*

2.4 Authorship Obfuscation for Deepfake Texts

In the task of DTD, AA models are evaluated under adversarial settings to assess their robustness. Due to the security risk, DTGs pose, it is important that AA models are robust to adversarial perturbations. The problem of administering adversarial perturbations to texts to cause an accurate AA model to assign inaccurate authorship is called **Authorship Obfuscation** (AO). This is because AO is the process of masking an author's writing style/signature to conceal the identity, usually for privacy reasons [100]. It is a strict case of Adversarial attacks [101], as the goal is to obfuscate writing style and preserve semantics, such that both human and automatic detection is evaded. See the differences between the two in Table 2.4. AO is a well-studied problem in the traditional AA community [100, 102–105], and has been extended to the niche AA community for DTD. See Table 2.5 for a detailed taxonomy of AA solutions for deepfake text detection and Figure 2.8 for the count of AO techniques, researchers have used to evaluate these AA models. This AO for deepfake texts problem is formulated as:

DEFINITION OF AO FOR DEEPFAKE TEXTS. *Given an AA model $F(x)$ that accurately assigns authorship of text t to k which can be either a human or a DTG author, the AO model $O(x)$ slightly modifies t to t^* (i.e., $t^* \leftarrow O(t)$) such that the authorship is disguised (i.e., $F(t^*) \neq k$) and the difference between t^* and t is negligible.*

Thus, we survey all AO techniques employed to obfuscate deepfake texts in different categories, as illustrated in Figure 2.7: *Stylometric Obfuscation*, and *Statistical Obfuscation*.

2.4.1 Stylometric Obfuscation

In order to build a robust stylometric classifier, as is observed in Section 2.3.1, an ensemble of features that capture several linguistic structures such as - Lexical, Syntax, etc. are required. However, to obfuscate an author's writing style, only one of the linguistic structures may be perturbed. Therefore, all the stylometric AO techniques only target a specific linguistic structure, unlike AA classifiers. Based on the stylometric obfuscation techniques used to obfuscate deepfake texts, we further divide this category into 4 categories - *Lexical, Syntactic, Morphological,* and *Orthographic* Obfuscation.

2.4.1.1 Lexical Obfuscation

Lexical relates to the word choice of a piece of text. Thus lexical obfuscation algorithms aim to mask authors' writing styles by replacing certain keywords with their synonyms while preserving semantics. Below, we discuss different techniques used to achieve lexical obfuscation for deepfake texts. Misspellings attacks may be considered a trivial AO technique, however, it is effective in obfuscation. The misspellings attack uses a list of commonly misspelled words[14] to determine which words to replace with their misspelled version. This AO technique is successful in obfuscating texts generated by GPT-2 and GROVER, and thus, evades detection of the following AA models - *GLTR*, *GROVER*, and *GPT-2* detector [53]. *GROVER detector* is further evaluated with this AO technique by obfuscating texts generated by GROVER. With only less than 10% of the texts perturbed, this attack is 94% successful [89]. However, this attack can be maneuvered by spell check algorithms, making the obfuscation technique, not robust [53]. In addition to misspellings, a whitespace attack ("will face" $\rightarrow$ "willface") is used to evaluate the robustness of *GROVER detector*. With only less than 4% of the texts perturbed, the attack is 85% successful [89].

Interesting artifacts/characteristics of deepfake texts still remain somewhat elusive. Therefore, perturbing these deepfake texts could reveal characteristics that have evaded the AA & AO community. Hence, using linguistic and statistical perturbations of words in the text, [57] extract important characteristics of deepfake text. For the linguistic-based perturbations, a lexical obfuscation technique is implemented - *Deduplicate tokens* which keeps the first occurrence of a token/word as is and replaces other occurrences with *[MASK]* token. This AO technique surprisingly improves the AA performance, suggesting that reducing the number of token occurrences may remove trivial features, causing the AA classifiers to focus on the more important features [57]

Next, texts generated by GROVER are obfuscated with the following techniques: (1) *varying sentiment*: changing the sentiment of words by replacing the word with another word of a different sentiment (positive $\rightarrow$ negative); and (2) *entity replacement*: replace entity with a useless entity [56]. Results suggest that both *GROVER* and *GPT-2 detectors* are vulnerable to these lexical-based perturbations.

Mutant-X [100] and Avengers [106] are used as baseline AO techniques to obfuscate the TuringBench dataset [59]. Mutant-X uses a genetic algorithm to search for suitable word replacement such that the semantics are preserved and the internal/substitute AA model misclassifies [100]. This process is notorious for its expensive computational complexity [107]. An internal model is used because, in the real world, the original AA model may not be known. Moreover, Mutant-X generates the obfuscated text and tests if it has evaded the AA model. If evasion is not successful, the process is repeated and tested for the defined max number of iterations [100]. These factors significantly

[14]`https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines`

increase the runtime of Mutant-X. Furthermore, the success of Mutant-X is dependent on how strong the internal AA model, which undermines the generalizability of Mutant-X. Hence, Avengers [106], an improved version of Mutant-X is proposed. Avengers is an ensemble version of Mutant-X. Unlike, Mutant-X, the internal AA model is an ensemble AA model, such that each classifier out of N classifiers focuses on different linguistic structures - syntax, semantics, etc.

DeepWordBug [108], a realistic character-level black-box attack is used to evaluate the robustness of 3 types of model - Statistical classification model [109], RoBERTa [110], and an Ensemble model (Statistical model + RoBERTa) [54]. It perturbs characters such that misclassification is maximized, while the Levenshtein edit distance is minimized [54, 108]. These models were trained with GPT-2 medium webtext and tested 3 separate test datasets - human vs. neural webtext from GPT-2 medium, GPT2 XL, and GPT-3 [54]. While deep learning-based classifiers achieve a higher performance in unperturbed/clean texts, statistical classifiers were found to be more robust to obfuscation [54]. Thus, the Ensemble model merges the advantages of high performance and adversarial robustness of the 2 models. DeepWordBug did not reasonably degrade the Ensemble model's performance, suggesting that DeepWordBug is not robust for this task. However, when DeepWordBug is used to evaluate the robustness of *GROVER detector* (Mega model) and *OpenAI detector* (base and large models) by perturbing the GPT-2 generated Yahoo answers & Yelp Polarity vs. Human datasets, it is successful [25]. In fact, DeepWordBug is found to be a very successful AO technique, reducing the accuracy of the Yahoo and Yelp datasets from 67.9% to 0.4% and 87.4% to 6.9%, respectively [25]. This suggests that the *OpenAI* and *GROVER detectors* are not robust to this kind of AO technique when evaluated on GPT-2 generated Yahoo answers & Yelp Polarity.

In addition, TextFooler [111], a realistic word-level black-box attack is used to evaluate the robustness of the Statistical model, RoBERTa, and Ensemble model (Statistical model + RoBERTa) [54]. TextFooler replaces words with synonyms based on cosine similarity within the embedding space [54, 111]. Based on the results, TextFooler is a robust AO technique, especially for Transformer-based models. Furthermore, as a substitute for human judgment, *MAUVE* is used to measure the human judgment of obfuscated texts. [54] finds that adversarial perturbation reduces *MAUVE* score which means that text quality is degraded and therefore deepfake texts are likely to be detected accurately by humans.

To further evaluate the robustness of the AA models - *GLTR* (*GLTR-BERT* & *GLTR-*

GPT2), *GROVER detector*, *BERT-Defense*, *FAST*, and *RoBERTa-Defense*, texts generated by GPT-2 and GROVER are obfuscated with TextFooler, Random Perturbations [55], and DFTFooler [55] AO techniques. Random Perturbations is an attack method that replaces random words with synonyms while preserving the semantics. DFTFooler is similar to TextFooler but only needs a publicly available pre-trained LM to generate obfuscated texts. This makes DFTFooler not as computationally costly as TextFooler [55]. Also, DFTFooler perturbations are transferable [55]. To find a valid word substitution using DFTFooler, there are 4 steps: (1) synonym extraction; (2) POS checking; (3) Semantic Similarity checking; and (4) Choose a synonym with low confidence as measured by a LM. BERT and GPT-2 XL are used as the LM for DFTFooler. Results suggest that TextFooler is a stronger AO technique than DFTFooler, but performs comparably, achieving 23-91% Evasion Rate [55]. Evasion rate is defined as the fraction of perturbed deepfake text that evades detection by an AA model. A high evasion rate indicates a high attack success. Furthermore, using a bi-directional LM (BERT) as the backend for DFTFooler, and increasing the number of words perturbed, increases the evasion rate of DFTFooler. Based on the results, *FAST* is the most adversarially robust AA model. This may be due to the hybrid nature of the model as it combines the benefits of stylometric, statistical, and deep learning-based techniques as discussed in section 2.3. Another reason for *FAST*'s superior performance is its use of semantic features based on entities [55].

2.4.1.2 Syntactic Obfuscation

Syntax relates to the order of words in a piece of text. Thus, syntactic obfuscation techniques change the original arrangement of words in a document, in order to obfuscate the author's writing style. Below, we discuss such techniques used on deepfake texts. Characteristics of deepfake texts are extracted by perturbing the syntactic structure of the texts with the following syntactic perturbation techniques: (1) *Shuffle tokens* which randomly shuffles the word order of the texts; (2) *Retain only (non)-stopwords* which removes all words, except for stopwords [57]. The accuracy of the AA model only dropped marginally. This suggests that these AO techniques are not robust. It also implies that these syntactic features are not important for DTD.

The robustness of *GROVER detector* is further evaluated by syntactic obfuscation techniques on texts generated by GROVER. These techniques are: (1) *source-target exchange*: interchanging the source and target; (2) *alter numerical facts*: distort numerical facts; (3) *syntactic perturbations*: changing the word form by adding/removing punctuation ("There is" $\rightarrow$ "There's"); and (4) *article shuffling*: replacing $N\%$ of a real (human-written) article's sentences with N sentences of a fake (neural) article [56]. All AO techniques were successful, except *article shuffling*. Also, stylometric classifiers were found to be more robust, except when perturbed under stricter constraints, such as perturbing a large percentage of texts [56].

ALISON [59] is another syntactic AO technique. It reduces inference time by 100-200x when compared to SOTA AO techniques such as Mutant-X [100], and Avengers [106]. It has an internal classifier trained on a set of linguistic AA features, which allow ALISON to generate suitable phrase replacements that preserve semantics. ALISON is used to evaluate the robustness of 3 Transformer-based models - BERT, DistilBERT, and RoBERTa by obfuscating 2 datasets - TuringBench and Blog Authorship Corpus [59]. It is able to obfuscate the datasets well, causing the AA models to underperform on obfuscated texts. Furthermore, ALISON is able to preserve the semantics of the original text much better than their baseline AO techniques (i.e., Mutant-X and Avengers).

2.4.1.3 Morphological Obfuscation

Morphology is the study of word forms. Thus, morphological obfuscation techniques change the configuration of a word (e.g. "won't" $\rightarrow$ "will not"). Upper/Lower Flip ("Leaving" $\rightarrow$ "LeavIng") is a morphological AO technique that may be considered trivial. However, it is successful in obfuscating texts generated by GROVER which significantly reduces the performance of *GROVER detector* [89]. With only about 2% of the texts perturbed, it achieved a 96% success rate in evading *GROVER detector*'s detection [89].

2.4.1.4 Orthographic Obfuscation

Orthography is the spelling convention of a language. Thus, orthographic obfuscation techniques aim to change the original spelling convention used in a piece of text to mask an author's writing style. Below, we discuss such techniques. Homoglyph attack is an orthographic perturbation technique that changes the unicode of texts. It changes English characters to cyrillic characters. This attack is almost imperceptible to the human eye and therefore, preserves semantics. The robustness of *GPT-2 detector*, *GROVER detector*, and *GLTR* is evaluated by obfuscating texts generated by GPT-2 with the homoglyph attack. *GPT-2 detector*'s performance dropped from 97.44% to 0.26% Recall. The perturbed texts caused *GROVER detector* to grossly misclassify deepfake texts as human-written texts and *GLTR* to shift the distribution (i.e., color scheme) of the perturbed texts [53]. *GROVER detector* is further evaluated on obfuscated texts generated by GROVER [89]. Homoglyph attack achieves a 97% success rate in perturbing *GROVER detector* [89]. However, this AO technique can easily be rendered ineffective by using spell check algorithms [53].

2.4.2 Statistical Obfuscation

In order to extract statistical characteristics from deepfake texts, the following statistical AO techniques are implemented: (1) *Replace text with likelihood ranks*; (2) *Replace text with specific linguistic features, such as Part of Speech, Dependency Trees, Constituent Trees* and *Named Entities*; and (3) *Retain tokens in high/low frequency regions* which defines a frequency gap score to calculate and extract the high and low-frequency words in the text [57]. The following 3 datasets are perturbed - human-machine pairs, Writing Prompt dataset [112], and CnDARIO (Chinese Novel Dataset crAwled fRom mIxed online sOurces) generated with GPT-2 fine-tuned with Chinese Literature. These datasets are in 3 different domains, respectively - Online Forum, News, and Literature. Results suggest that the *high/low frequency region* perturbations is the most effective obfuscation technique [57]. This further suggests that the *high/low-frequency region* feature could be an effective feature for distinguishing deepfake texts from human texts.

ruBERT for DTD is evaluated on 2 AO techniques - (1) *calculating the class probabilities of each label and only selecting the texts with the highest human class probability*; (2) *adds the detector's log-probability to the beam search decoding strategy during generation* so that only more human-like texts are generated [28]. These attacks achieve 46% and 56% success rates, respectively. *RoBERTa-Defense* is evaluated by changing the sampling distribution of the deepfake texts in the test set. This obfuscation technique involves: (1) *varying the text decoding strategy (and its parameters)*; and (2) *varying the number of priming tokens* [55]. The quality of the obfuscated deepfake texts is measured by GRUEN [113], a metric used to measure the linguistic quality of AI-generated texts (deepfake texts). GRUEN has a high correlation with human judgments. The score ranges from $0 - 1$, and a higher value indicates high linguistic quality. Linguistic quality is based on grammaticality, non-redundancy, discourse focus, structure, and coherence [55]. Using GRUEN, a *successful attack* is defined as an attack that degrades the performance of the AA models, with little to no linguistic quality (GRUEN score) degradation [55]. The results suggest that changing the decoding strategy is an effective AO technique. Even *GLTR*, a statistical AA model is fooled by this AO technique [55].

2.5 Evaluation of AA/AO Methods

2.5.1 Machine-based Evaluation

2.5.1.1 Authorship Attribution

To evaluate AA models, literature often used popular classification metrics such as *Precision*, *Recall*, *Accuracy*, and *F1 score*. For instance, [53] used the recall metric to evaluate the robustness of AA models toward AO techniques. Previous works evaluate the generalizability of AA models, not only on a standard single test set [19,30,33,52,55] but also on several out-of-sample distributions [25]. For example, [25] evaluate *GROVER detector* (Mega model) and *OpenAI detector* (base and large models) on three variants of test sets, namely *in-distribution, out-of-distribution* and *in-the-wild* datasets. *In-distribution* datasets are test sets sampled from the same distribution of the training set, while *Out-of-distribution* datasets are those sampled from a different distribution from the training set. This test dataset is created using PPLM [11] and GeDi [51] to control the GROVER and GPT-2 generations [25]. The *in-the-wild* datasets are test sets generated from a DTG, different from the DTG used to generate a training set [25]. To build this *in-the-wild* dataset, training sets contain texts generated from GPT-2 and GROVER pre-trained models, while test sets contain texts generated from GPT-3 and a fine-tuned GPT-2 model [25]. In general, AA models perform well on *in-distribution* datasets, but suffer on *out-of-distribution* datasets, and even more on *in-the-wild* datasets.

2.5.1.2 Authorship Obfuscation

To evaluate AO models, literature often uses the *success rate* [28, 89], a fraction of successfully obfuscated (i.e., misclassified) texts which were accurately attributed prior to obfuscation. Another measure for AO models is *evasion rate* [55] which is the fraction of perturbed deepfake texts that evade the detection by an AA model. Next, due to the time and financial cost required to carry out a human-based evaluation, *MAUVE*, a metric that statistically emulates human judgments in terms of the linguistic quality (i.e., coherency) of neural texts vs. human-written texts, has been used on the AO problem [54]. That is, *MAUVE* is used as a substitute for human evaluation of obfuscated text vs. non-obfuscated text [54]. Furthermore, based on the strict definition of AO, it is sometimes important that the obfuscated text preserves the semantics of the original text. Hence, literature has measured the degradation of semantics between original and obfuscated texts, namely METEOR [114], Universal Sentence Encoder (USE) [115] Cosine similarity, and GRUEN [113]. These metrics all correlate with human judgments and a high score indicates an obfuscated text with well-preserved semantics.

2.5.2 Human-based Evaluation

There is currently no known human-based evaluation of AO models. The closest one is the machine-based simulation by MAUVE [54]. As such, in this section, we focus our review on the human-based evaluation of AA models, especially in the context of DTD.

2.5.2.1 Human Evaluation without Training

A set of research works recruited human participants (often from crowdsourcing platforms such as Amazon Mechanical Turk (AMT)) and tested whether they can distinguish deepfake texts from human-written texts. A simple introduction to the tasks is given, but no special training is done for human participants in this line of research. For instance, [16] examined the quality of texts generated by GROVER vs. human-written texts by humans and found that humans find GROVER-written news more believable than human-written news. [116] asked human participants to detect infilled texts filled by deepfake texts (e.g., she drank [blank] for [blank]) and found that humans had difficulty in detecting the infilled texts filled by deepfake texts. [1] introduced a benchmark for AA research, *TuringBench*, evaluated the performance of humans in distinguishing 19 pairs of human vs DTG (e.g., human vs. GPT-1 or human vs. FAIR) using TuringBench, and found that humans on average scored 51-54% of accuracies, only slightly better than random guessing. Unsurprisingly, [31] also found that humans were unable to accurately detect GPT-3 texts from human-written texts. [33] evaluated the quality of top-p and top-k decoding strategies, and found that (1) AA models detect deepfake texts generated by top-k decoding better than humans, but (2) humans detect deepfake texts generated by top-p decoding better than AA models. Lastly, [36] found that both humans and AA models struggled to detect neural fake reviews.

2.5.2.2 Human Evaluation with Training

Another line of research attempted to first train human participants about DTD tasks and measure the performance improvements afterward. For instance, when human participants were trained to use GLTR [2] in detecting deepfake texts, thanks to the color scheme of GLTR (see Figure 2.6), human performance increased from 54% to 72% in accuracy. To further evaluate deepfake texts, [117] proposed a framework to collect a large number of human annotations via a game, Roft[15], on the quality of neural vs. human texts. Human participants were told to detect the boundary at which an article transitions from human-written to AI-generated. Only 16% of human participants were able to correctly identify the accurate boundaries [117]. [34] studied three training strategies–instruction-based, example-based, and comparison-based, and found that example-based training is the most effective to improve human performance for solving DTD tasks (achieving the average accuracy of 55%) across the domains of story, recipe, and news. Next, [23] investigated how accurately humans can classify real vs. generated articles with and without images, using different types of news datasets: real captions and articles, real captions, and generated articles, generated captions and real articles and generated captions and articles. By conducting an AMT-based study, untrained and trained human participants were able to achieve an average of 46% and 68%, respectively. Further investigation on the trustworthiness of the different article types based on style, content, consistency, and overall trustworthiness reveals that humans were skeptical about the overall trustworthiness of news articles across all four types [23]. Finally, recently, [118] proposed a framework for scrutinizing deepfake texts through crowdsourced data annotation in a scalable fashion, where deepfake texts were shown to have various error types: language-errors (i.e., lack of coherency and consistency in text), factual-errors, and reader-issues (i.e., text is too obscure or filled with too much jargon so that understanding is negatively impacted).

In conclusion, literature has found that humans alone cannot detect deepfake texts accurately, achieving detection accuracies only slightly better than random guessing. When humans are properly trained about the characteristics of deepfake texts, further, this detection accuracy tends to increase but only by small margins.

[15]http://roft.io/

2.6 Applications

Deepfake Detection: Successful solutions for AA/AO tasks can be useful in many applications. For instance, recently, the generation of realistic AI-made images[16] and videos, so-called “deepfakes”, have flooded the Web. While most of these deepfakes images/videos are made for humor, some are malicious in generating misinformation, spreading political propaganda, or attacking individuals [119]. In literature, in particular, [23] studies the realistic scenario where real images would be paired with deepfake texts to increase the authenticity of a news article as well as evade detection. In such a setting, successful AA solutions can point out the non-human nature of deepfake texts to users or can be used to extract features of deepfake texts for downstream deepfake detection models.

Chatbot Detection: Another application is for AA solutions to detect suspicious messages (e.g., phishing or chatbot messages) that may have been (partially) generated by DTG. Similar to deepfake texts of news format, shorter or informal chatbot messages are also hard to discriminate when generated by machines [120, 121]. An example of a state-of-the-art chatbot is ChatGPT [122] which has been used to generate medical writings [90, 123], finance writings [124], etc. These applications of ChatGPT have also increased the likelihood of cheating in academic writing [125]. Thus, AA models for deepfake text detection will be beneficial in distinguishing chatGPT-generated texts from human-written texts.

Anonymity Preservation: On the other hand, successful AO solutions can be used to help individuals who have needs to share their writings without jeopardizing their secret identity. For instance, an NGO activist or whistleblower may submit her op-ed to news media after making sure that no popular AA solutions can attribute the writing to her.

2.7 Summary

With the rise of deepfake texts that were generated by large-scale language models, we are currently in an arms race between generation and detection of *deepfake texts*. In this work, we comprehensively survey two important problems of deepfake texts: **Authorship Attribution** (AA) and **Authorship Obfuscation** (AO). We first categorize existing AA

[16]https://thispersondoesnotexist.com/

solutions into four types of stylometric, deep learning-based, statistical, and hybrid attribution. Similarly, we categorize existing AO solutions into two types of stylometric and statistical obfuscation, and elaborate pros and cons of representative methods therein. In addition, we discuss different evaluation methods for AA and AO problems in the context of deepfake texts, and finally, share a few important challenges that we feel lacking currently. By and large, we believe that the data mining community is well-positioned to be able to contribute to significant improvement in both AA and AO research. Despite their close implications in security and privacy, with respect to the underlying methods used, their problem formulation as supervised or unsupervised learning, and their focus on the accuracy and running time as major metrics. Lastly, to mitigate the challenges of accurate detection of deepfake text, [126] proposes watermarking these text-generative language models. This entails embedding humanly imperceptible signals into the language models such that they generate semantically relevant texts, unnoticeable to humans but noticeable to detectors. These watermarking [126, 127] techniques attempt to solve the security risks that these language models pose. However, as these watermarking techniques have not yet been widely adopted, we still have to rely on AA and AO solutions for deepfake text detection. Also, as such watermarking techniques are a recent/new development, their robustness to strong AO techniques has not yet been evaluated.

Chapter 3 | Authorship Attribution for Deepfake Text Generation

3.1 Introduction

As novel NLG techniques become more sophisticated and prevalent, corresponding pitfalls and risks of such technologies also increase. Adversaries may use such technologies to generate realistic artifacts to trick naive users into fraudulent activities (e.g., chatbot conversation in a phishing scam or deepfake-based disinformation campaign). Therefore, the need to distinguish deepfake-generated texts from human-written ones, so-called the *Turing Test*, naturally arises. Furthermore, in some security applications, merely being able to identify deepfake-generated text may not be sufficient. Instead, a more critical solution would be to tell *which* NLG method among many candidates has generated a given text in question–so-called the *Authorship Attribution (AA)* problem. To improve our understanding of this newly-emerging problem, we empirically investigate three versions of the AA problem in this paper. For all three versions, we assume that there are k different NLG methods[1].

Problem 1 (**Same Method or Not**). *Given two texts T_1 and T_2, determine if both T_1 and T_2 are generated by the same NLG method (or human) or not.*

Problem 2 (**Human vs. Deepfake**). *Given a text T_1, determine if T_1 is written by a human or generated by any k NLG methods.*

Problem 3 (**Authorship Attribution**). *Given a text T_1, single out one NLG method (among k alternatives) that generated T_1.*

[1]In the future, we envision that k can be huge, say 1000, but for this experiment, we set k=8.

We model P1 (Problem 1) and P2 (Problem 2) as the binary classification problem, while P3 (Problem 3) as the multi-class classification problem. All three problems are related, with several motivations as follows.

First, solutions to P1 may be useful when one needs to determine the plagiarism or identity theft issue of an NLG method. For instance, suppose GPT-2 becomes very powerful in the near future so that other NLG methods may even try to mimic the characteristic features in the GPT-2-generated texts. Then, a solution to P1 can determine if two texts in question are both generated by GPT-2 or not. Second, as NLG methods become ubiquitous, the threat of generating misinformation at scale increases naturally. Thus, using solutions to P2, accurately distinguishing between deepfake- and human-generated texts is required to mitigate the security risks that NLG methods could pose. Finally, as to P3, as the number of state-of-the-art NLG methods increases, it will be beneficial not only just to separate them into two camps but also find out which generators are used. Furthermore, knowing each generator's writing signature or style moves us closer to quenching the security threats that they may introduce.

3.2 Related Work

3.2.1 Features for Authorship Attribution

Predicting an author based on their writing signature is called Authorship Attribution (AA). This AA problem has been previously and even recently solved with n-grams [84, 128–133]. Next, as complex datasets emerge, other techniques such as POS-tags [134–136], topic modeling (i.e. LDA, AT and DADT) [137–139], POSNoise [140] and LIWC [133, 141] are explored and used to solve the AA problem. However, [134] claim that n-grams and POS-tags are not sufficient for solving the AA problem, and sometimes negatively impact classifiers' performance. Therefore, they recommend using discourse embedding features.

Furthermore, [142] attempts to solve the AA problem with online messages by investigating four types of writing-style features (i.e., lexical, syntactic, structural, and content-specific features). Structural and content-specific features were the best to assign authorship [142].

Next, [131] examines the use of content and function words as relevant features for AA. [143] focuses on content words and so parse phrase-structures. Consequently, [144] claim that word pairs could make strong features and extract function words, content words, similarity and relatedness. They find function words to be the most robust feature [144]. To establish distinct writing styles further, [145] extract lexical, syntactic, and stylistic features using bag-of-words (freq. of unigrams), POS-tags, Dependency relations, and Chunks (unigram freq.), respectively for the AA problem. With the new wave of nuanced techniques to solve the AA problem, [146] use syntax tree for each sentence to analyze grammar. Finally, [121] use readability scores to distinguish human-written texts from deepfake texts, another form of AA. The results suggest that readability is a vital feature for distinguishing authors.

3.2.2 Classifiers for Authorship Attribution

Several well-established classical machine learning classifiers have been applied to the AA problem, including Naive Bayes [147, 148], SVM [121, 145, 149], Conditional Tree [128], Random Forest [128, 149, 150], and KNN [150]. However, due to improvements in neural networks, recently, CNN (Convolution Neural Network) is said to be even more suited for the AA problem [134, 136, 151]. CNN architecture is better suited to represent the characteristics of each author. Consequently, [134] improve the CNN's use with discourse features (i.e., n-grams and POS-tags). [152] further improve upon CNNs with the use of word embeddings to represent texts. Lastly, RNNs have also been shown to be well-suited for representing the authors' distinct writing styles [153].

3.2.3 Applications of Authorship Attribution

The applications of AA are vast and include: assigning authorship to literature/text, and ascertaining the demography of an author (e.g., age, gender, native language) [154]. AA can also be applied to predicting author(s) of source code [155], chatbot detection [141], and even detecting authors intentionally trying to mask their writing style [156, 157]. Finally, our work bears similarity to [158], which investigates the stylistic properties of different neural text generation techniques (i.e., Ngram-based and RNN-based).

3.3 Generation of Texts

We have nine text generators–i.e., one human writer and eight neural deepfake generators. All eight neural generators require a short prompt to begin their generation and the

PROMPT	Putin and Xi are using the coronavirus crisis to extend their control. Across the world, Trump is struggling to keep up
Human	… Donald Trump's shock announcement shutting down travel from much of Europe came with the customary back-pedaling common to many of his hastily announced initiatives. At first, Trump said trade and travel would be targeted, in his latest attempt to stem the spread …
CTRL	…. President Vladimir Putin, meanwhile, says there was no reason why Moscow would spread such diseases. But both leaders have been unable so far to … [read more] what do you think about this story? do we need another Ebola scare or two? …
GPT-1	… with them in a war that has been going on for decades now - but he's still not strong enough. " " i'm sorry about your family. it must be hard being so far away from home. " she said softly. " yes, we're all very lonely here. there wasn't much time before …
GPT-2	… Putin's paranoia over the contagion of Ebola is a common theme. As his recent visit to Washington shows, he's playing a dangerous game. And here's what Putin wants Putin wants to use Ebola as a pretext for imposing further Russian sanctions on the West. …
GROVER	… hours after world cup workers hand-picked by Putin to fix problems faced by workers in turkey and Venezuela, a new threat to trump is receding.According to Wikileaks, the Coronavirus outbreak is receding and the Russian leader, Putin,…
XLM	… with Russia alone ruling over Europe while china under Mao Zedong rules Asia in turn which makes it difficult for him politically but financially as well being president / CEO of America first bank & chairman's nominee at CIA …
XLNET	… with china or Russia in his efforts at de-destructuring America from the financial crisis that has taken place over the past two years of this century as well a global economic downturn that will continue for many more months into next year …
PPLM	… the latest developments in Syria, Ukraine, Russia's intervention in the middle east and china's rise to global power. In Moscow, a group of Russian diplomats, who did not want their real names to be reported on the record…
FAIR	.. He has ordered his federal agencies to step up efforts to combat it and to warn of new threats. in Washington, Trump's national security adviser, h.r. mcmaster, said on Monday the U.S. government had seen signs of the bug and that countries in the Middle East were on high alert. …

Table 3.1: Snippets of nine texts using the titles of human-written articles as the prompt to neural methods.

number of words to generate. These eight generators were chosen because we found that they had the best pre-trained models for our task. We used the titles of news articles (written by human journalists) as the prompt and set 500 as the number of words.

1. **Human**. We collected recently-published news titles and contents in mostly Politics–819 from CNN, 132 from Washington Post, and 113 from the New York Times. As professional reporters write these news articles, they represent human-written texts. Then, we used the news titles as the prompts for other neural methods.

2. **CTRL**. Also known as "Conditional Transformer Language Model For Controllable Generation," CTRL[2] is a huge language model with 1.63 billion parameters [10]. The model was trained on control codes to guide the styles and contents

[2]https://github.com/salesforce/ctrl

Measure	Human	Deepfake							
		CTRL	GPT-1	GPT-2	GROVER	XLM	XLNET	PPLM	FAIR
# of samples	1,066	1,066	1,066	1,066	1,066	1,066	1,066	1,066	1,066
AVG word count	432.31	530.03	345.03	199	356.76	441.32	452.58	228.89	250.42
SD word count	270.82	73.51	10.79	74.15	114.96	34.67	32.59	64.13	39.94
AVG sentence count	26.87	33.02	32.64	15.68	21.64	3.97	5.02	13.53	17.53
SD sentence count	19.49	21.18	5.55	6.99	9.65	1.71	1.97	4.61	4.88

Table 3.2: Summary statistics of nine generated texts (one by human and eight by neural methods).

of generated texts. Among the 50 control codes available, we used the *News* control code to generate long articles.

3. **GPT**. The OpenAI GPT-1 is built with Transformers. It was trained and modeled after a simple concept - to predict the next token, given the previous token [7]. We used the medium GPT-1 model with 345 million parameters since it was computationally less expensive while still being able to generate comparable results. We used the Transformer text generation setup by huggingface[3].

4. **GPT-2**. We also used the GPT-2 model with 774 million parameters. We used the GPT-2 wrapper[4] to generate texts.

5. **GROVER**. Grover is another large language model, explicitly trained to generate political news [16]. It uses the same template as news outlets such as CNN and the New York Times. Grover uses the same architecture as GPT-2 and the same concept of predicting the next token, given previous tokens. We used code from repo[5] to generate texts.

6. **XLM**. The Cross-lingual Language Model (XLM) is another generative language model [39]. Unlike other language models, XLM is trained for the task of cross-lingual classification. We generated texts from the English language model, using the same setup in huggingface as GPT.

7. **XLNET**. XLNET [40] improves language modeling by introducing bidirectional contexts. This technique involves a generalized auto-regressive pre-training method

[3]https://github.com/huggingface/transformers
[4]https://github.com/minimaxir/gpt-2-simple
[5]https://github.com/rowanz/grover

Measure	Human	Deepfake								AVG
		CTRL	GPT-1	GPT-2	GROVER	XLM	XLNET	PPLM	FAIR	
Flesch Reading Ease	37.97	60.97	68.68	54.49	46.63	46.40	48.94	44.97	51.85	51.21
Flesch-Kincaid Grade	12.79	9.58	8.48	10.27	11.53	11.64	11.28	11.66	10.76	10.89
LIWC-Authentic	25.3	54.28	61.66	15.1	23.76	48.06	80.69	34.27	18.77	40.21
LIWC-Analytic	89.81	51.99	40.93	92.59	89.98	78.61	50.46	73.18	92.89	73.38
LIWC-Article	7.98	1.47	3.18	11.87	8.69	0.59	2.03	2.6	10.05	5.38
Entropy	7.81	8.98	8.01	6.52	7.79	8.99	8.91	7.77	7.41	8.02

Table 3.3: Linguistic features of nine generated texts. The AVG column is the average of each row.

and adopts the Transformer-XL framework into pre-training. XLNET achieved state-of-the-art results, outperforming BERT on 20 tasks. We also used the huggingface Github repo to generate texts.

8. **PPLM**. The "Plug and Play Language Models (PPLM)" is another language model that improves upon GPT-2 by fusing the medium model with bag of words models [11]. We used the *Politics* bag of words model to generate texts', using the code[6], and used the perturbed version.

9. **FAIR**. Facebook's FAIR has three language models of English, Russian, and German [8]. For our task, we used the English language model built with FAIRSEQ sequence modeling toolkit[7].

Table 3.1 shows the snippets of nine texts for the identical prompt message–i.e., one written by news reporters and eight by different neural language models. Table 3.2 shows the summary statistics of nine generated texts. Note that CTRL tends to generate the longest texts (in terms of the number of words), while GPT-2 tends to generate the shortest texts. Both XML and XLNET generated the texts with very long sentences.

3.4 Linguistic Analysis

We first conduct a psycholinguistics study to analyze different linguistic features of generated texts. The result is summarized in Table 3.3.

[6]https://github.com/uber-research/PPLM
[7]http://shorturl.at/swDHJ

First, we use *Flesch Reading Ease* and *Flesh-Kincaid Grade* to gauge generated texts' readability. *Flesch Reading Ease* generates a score between 0 and 100, such that post-college level yields a score between 0-30, college-level yields 31-50, high-school level yields 51-70, middle school yields 71-90, and 5-th grade level of reading and below yields 91-100. These seven reading levels also go from a scale of very-difficult-to-understand due to the level of sophistication to very-easy because it is the grade level of readability. Therefore, obtaining a post-college level (i.e., low score) is uncommon and impressive if a deepfake text-generator generates such texts. On the other hand, the *Flesh-Kincaid Grade* generates a score representing the U.S. grade level of education (the higher, the more sophisticating). For instance, text given a 10.8 score suggests that its author can be in the 11-th grade and about 16-17 years old.

Next, we use *Linguistic Inquiry and Word Count (LIWC)* [76] to capture the psycholinguistics features. LIWC has 93 features, of which 69 are categorized into: Standard Linguistic Dimensions (e.g., pronouns, past tense), Psychological Processes (e.g., social processes), Personal concerns (e.g., money, achievement), and Spoken Categories (e.g., assent, nonfluencies) [141]. Table 3.3 includes top-3 distinguished LIWC features among all generation methods. A high *LIWC-Authentic* score means that the author of the text is honest or less evasive. We can observe that GPT-1 and XLNET generates more personal content than GPT-2 and FAIR. *LIWC-Analytic* reflects the formality, and logical nature of the text. GPT-2, GROVER, and FAIR scores are as high Human, suggesting that they all generate sophisticated texts. *LIWC-Article* shows the usage of *a, an, the*, which are crucial in any formal writing. Similar to *LIWC-Analytic*, GPT-2, GROVER, and FAIR scores are similar to Human. Overall, the patterns among these LIWC features follow our observations that GPT-2, GROVER, and FAIR generally have higher news generation quality than other deepfake algorithms. Finally, we also measure the entropy scores of generated texts [159]. Figure 3.1 shows the 2-dimensional distribution of generated texts using Principal Component Analysis (PCA) on all psycholinguistic features, with about 70% explained variation. As we can observe a large overlapped portion among generated texts. We expect a non-linear machine learning model (e.g., Random Forest) would perform better than a linear method such as Naïve Bayes in classifying the texts according to their generators using these features.

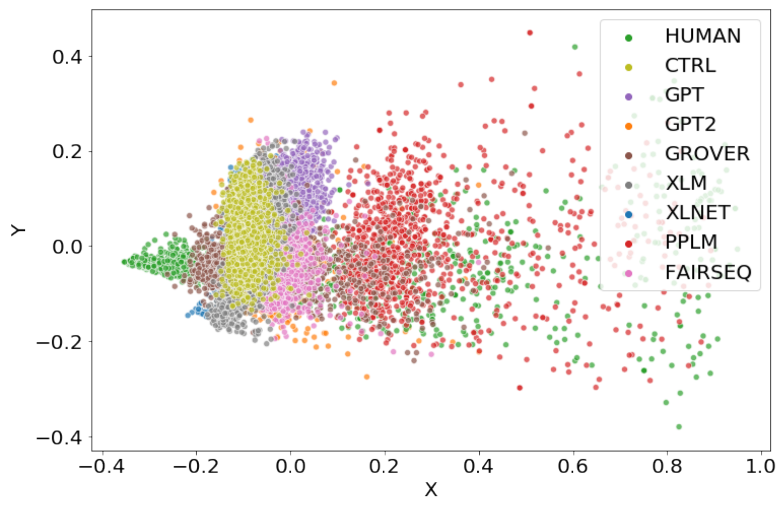


Figure 3.1: Distribution of generated texts on 2-dimensions using PCA of Linguistic features - LIWC, Readability, & Entropy.

3.5 Model Architecture

In solving three problems, we compare various relatively-simple neural models' performances, employing different architectures to encode generated texts into representation vectors, which then feed into a fully connected network followed by a *softmax* layer for prediction. Note that our goal is *not* to develop sophisticated neural models to solve three problems. Rather, we want to empirically evaluate how these simple neural models (as baselines) perform in solving three problems.

1. **Embedding:** This model maps each word in the generated texts to a vector of 300 dimensions, then sums up all resulting vectors as the final representation.

2. **RNN:** This model uses a variant of recurrent neural network (RNN) with a GRU [160] layer to model the sequential dependency among words within each of the generated texts.

3. **Stacked_CNN:** This model is inspired by [161], where each of generated texts is encoded by a sequence of six 1D convolutional layers of different kernel sizes. We reduced learning rates from 0.001 to $0.01/0.1$.

4. **Parallel_CNN:** Similar to *Stacked_CNN*, but instead of using a stack of convolutional layers, we adopt [162] and use four parallel 1D convolutional layers of

Model	Balanced (1:1)			Imbalanced (1:8)		
	P	R	F1	P	R	F1
Embedding	0.9006	0.8683	0.8841	0.5148	0.7531	0.6116
RNN	**0.9748**	**0.9879**	**0.9813**	0.5439	0.8695	0.6692
Stacked_CNN	0.9509	0.9747	0.9626	0.6269	**0.9269**	0.7479
Parallel_CNN	0.9545	0.9852	0.9696	0.6004	0.8319	0.6974
CNN-RNN	0.9572	0.9750	0.9660	**0.6847**	0.9248	**0.7869**

Table 3.4: P1: Binary classification performance of "Same Method or Not" on two collective test sets. P is Precision, R is Recall, and F1 is the Macro F1 score.

different kernel sizes, followed by a max pooling and concatenation operation.

5. **CNN-RNN:** This is a combination of *Stacked_CNN* and *RNN* where each word of a text is first encoded by a stack of two 1D convolutional layers before being input into each step of a GRU layer to model the sequential dependency of the whole text.

Experimenting with these neural models, we split the dataset into the training, validation, and testing parts in 7:1:2 ratio.

3.6 P1: Same Method or Not

The first version of the problem is to determine whether two given texts are generated by the same method (including human writers) or not. Even if one cannot pinpoint whom the author is for a given text, one may still notice similarities between texts. Therefore, P1 tests the varying capabilities of models to detect such *similarities* between the two texts.

We prepare two datasets of a similar size. In the balanced set, half of text pairs are generated by the same method (e.g., Human-Human or CTRL-CTRL), and the other half are random pairs of the two different methods (e.g., Human-CTRL or GROVER-FAIR). In the imbalanced set, 11% of text pairs are generated by the same method, while the remaining 89% are by different methods (1:8 ratio). Model-wise, we utilize the *Siamese neural network* [163] with one of the text encoders in Section 3.5 to predict whether the two input texts are generated by the same method. Table 3.4 summarizes the performances. Both RNN and CNN-RNN methods perform the best in the balanced and imbalanced settings, respectively. Recall that the imbalanced setting is more challenging

Model	CTRL	GPT	GPT2	GROVER	XLM	XLNET	PPLM	FAIR	AVG
Embedding	0.9768	0.9838	0.4044	0.6628	0.6535	0.6551	0.8449	0.5178	0.7124
RNN	**1.0**	0.9930	0.6329	**0.9977**	0.9977	**1.0**	0.9466	0.8812	0.9311
Stacked_CNN	0.9792	0.9815	0.6347	**0.9977**	0.9907	0.9186	0.6457	0.6316	0.8475
Parallel_CNN	**1.0**	**0.9977**	0.6075	0.9536	**1.0**	**1.0**	0.9513	0.9282	0.9298
CNN-RNN	**1.0**	0.9861	0.6626	**0.9977**	0.9699	0.9907	0.7949	0.7018	0.8880
RoBERTa	0.6448	0.6404	0.6407	0.6448	0.6490	0.7185	0.6404	0.6404	0.6524
RoBERTa-tuned	0.9730	0.9881	**0.9792**	0.8894	0.9921	0.9850	**0.9796**	**0.9753**	**0.9702**
GROVER-DETECT	0.7753	0.7319	0.6976	0.8135	0.6929	0.7536	0.7761	0.7616	0.7503
AVG	0.9186	0.9128	**0.6574**	0.8696	0.8682	0.8777	0.8236	0.7547	

Table 3.5: P2: Binary classification performance in F1 score of "Human vs. Deepfake" on eight individual test sets. Each column name X indicates an individual balanced test set of HUMAN (50%) and X (50%).

Model	Balanced (1:1)			Imbalanced (1:8)		
	P	R	F1	P	R	F1
Embedding	0.4922	0.4877	0.4899	0.4555	0.5274	0.4770
RNN	0.7625	0.7611	0.7611	0.8242	0.6956	0.7390
Stacked_CNN	0.7592	0.7592	0.7592	0.6585	0.7252	0.6816
Parallel_CNN	0.9125	0.9118	0.9120	0.8370	0.8458	0.8413
CNN-RNN	0.7314	0.7315	0.7314	0.8198	0.7162	0.7546
RoBERTa	0.4949	**0.9540**	0.6517	0.1090	**0.9540**	0.1957
RoBERTa-tuned	**0.9196**	0.9109	**0.9152**	**0.9229**	0.7859	**0.8489**
GROVER-DETECT	0.8100	0.5590	0.6610	0.3337	0.5591	0.4180

Table 3.6: P2: Binary classification performance of "Human vs. Deepfake" on two collective test sets. P is Precision, R is Recall, and F1 is the Macro F1 score.

than the balanced as # of positive samples is much smaller. Overall, neural models can identify two texts generated by the same method very well for the balanced setting (F1=0.9813) and reasonably well for the imbalanced setting (F1=0.7869).

3.7 P2: Human vs. Deepfake

The second version of the problem determines whether a given text is generated by human or deepfake (i.e., one of the neural methods). P2 is a type of the *Turing Test*. Despite the recent advancements in neural NLG methods, we hypothesize that there may still be latent differentiating characteristics between human-written and deepfake-generated texts. Therefore, P2 tests the varying capabilities of different models to detect such *differences* between human and deepfake writings.

Model	Human	Deepfake								AVG
		CTRL	GPT-1	GPT-2	GROVER	XLM	XLNET	PPLM	FAIR	
Naïve Bayes	0.4668	0.9812	0.9835	0.4830	0.1901	0.9858	0.9810	0.9448	0.1812	0.6886
Decision Tree	0.7376	0.9835	0.9696	0.7239	0.6682	0.9837	0.9858	0.9626	0.5770	0.8435
SVM	0.8038	0.9953	0.9953	**0.8048**	0.7426	0.9953	**0.9976**	0.9742	0.6792	0.8876
Random Forest	**0.8122**	**1.0**	0.9953	0.7850	**0.8169**	**1.0**	0.9906	**0.9860**	0.7465	**0.9042**
Embedding	0.5727	0.9581	0.9688	0.7785	0.1080	0.9589	0.9026	0.7424	0.9900	0.7756
RNN	0.4190	0.9932	0.9906	0.7659	0.6295	0.9953	0.9929	0.8238	**1.0**	0.8456
Stacked_CNN	0.3415	0.9518	0.9638	0.7511	0.6603	0.9662	0.9104	0.8009	0.9950	0.8157
Parallel_CNN	0.5020	0.9790	0.9638	0.7579	0.6499	0.9976	0.9953	0.7582	**1.0**	0.8448
CNN-RNN	0.6366	0.9730	**1.0**	0.8038	0.5664	0.9813	0.9739	0.7942	**1.0**	0.8589
POS+CNN-LSTM	0.5868	0.6777	0.9109	0.7132	0.4798	0.8910	0.6845	0.8467	0.5689	0.7066
POS+LSTM-LSTM	0.2378	0.6746	0.8654	0.6512	0.4628	0.7572	0.6505	0.7520	0.5876	0.6266
3-grams + SVM	0.6992	**1.0**	**1.0**	0.6821	0.6579	**1.0**	0.9929	0.8165	0.6483	0.8330
Character n-gram + SVM	0.7008	**1.0**	**1.0**	0.6835	0.6534	**1.0**	0.9929	0.8114	0.6410	0.8314
AVG	**0.5423**	0.9360	0.9698	0.7218	0.5542	0.9633	0.9270	0.8366	0.7396	

Table 3.7: P3: multi-class classification performance with per-class macro F1 (for each column) and overall average F1 scores of models (for each row).

For P2, in addition to five neural models introduced in Section 3.5, we also tested three known Turing Test models including RoBERTa [110] using a similar implementation of *GPT-2 Output Detector*[8], GROVER-DETECT [16][9], and RoBERTa-tuned, which is the RoBERTa that we fine-tuned using 20% of our data. RoBERTa is fine-tuned by adding a classification layer on top of it. Next, the weight of the classification layer is randomly initialized and then trained on the GPT-2 output and human written text [10]. Further, we utilize the 20% of the target data we collected to fine-tune the RoBERTa classification model. Note that GROVER-DETECT used in our experiment was trained using only 5K training samples, while its improved version trained with 100K samples is not publicly available. Additionally, *GLTR* is another state-of-the-art Turing tester used to distinguish deepfake-generated texts from human-generated texts [2], although not used in these experiments.

Furthermore, in this setting, we tested both individual case (i.e., one neural method at a time) and collective case (i.e., eight neural methods combined). First, we prepare eight test sets for the individual case, each of which is the balanced test set between human (50%) vs. one neural generator (50%). Table 3.5 summarizes the performances in those eight individual test sets. For the collective case, on the other hand, we prepare two test sets. In the balanced set, the half of tests are written by human and the other eight neural

[8]https://github.com/openai/gpt-2-output-dataset/tree/master/detector
[9]https://github.com/rowanz/grover/tree/master/discrimination
[10]https://github.com/openai/gpt-2-output-dataset/

methods generates the other half. In the imbalanced set, 11% of test texts are written by human, while the remaining 89% are generated by any of the eight neural methods (1:8 ratio). Table 3.6 summarizes the performances in both balanced and imbalanced settings.

In Table 3.5, we find that GPT-2 generates texts that are almost indistinguishable from human-written texts (having the lowest average F1=0.6574 across eight models). FAIR is the second (F1=0.7547). Interestingly, we find that RoBERTa-tuned can still differentiate human-written texts from GPT-2-generated ones with a high F1 score (0.9792) and has the highest average F1 (0.9702) across all eight datasets. This is likely so because RoBERTa-tuned is fine-tuned on two doses of GPT-2 texts (i.e., RoBERTa was already fine-tuned on GPT-2 dataset to begin with).

For the performance of collective cases shown in Table 3.6, RoBERTa-tuned is again the overall winner. It can differentiate human-written vs. deepfake-generated texts with F1=0.9152 for the balanced setting and F1=0.8489 for the imbalanced setting. Two existing Turing Test models (i.e. GROVER-DETECT and RoBERTa) significantly underperform, although RoBERTa aces in Recall.

3.8 P3: Authorship Attribution

The third version of the problem is to single out the real author of a given text, among many alternatives (e.g., one human and k neural methods). Therefore, P3 tests different models' varying capabilities to exploit both *similarities* within and *differences* across human and deepfake writings.

For P3, in addition to five neural models introduced in Section 3.5, we also tested four classical machine learning models (i.e., Naïve Bayes, Decision Tree, SVM, and Random Forest) using psycholinguistic features discussed in Section 3.4 and four state-of-the-art AA solutions, including POS+CNN-LSTM and POS+LSTM-LSTM [85], 3-grams + SVM [164] and Character n-gram + SVM [165]. Neural methods such as Embedding, RNN, and CNN-RNN used GloVe word embedding [166], but Stacked_CNN and Parallel_CNN did not use GloVe due to its negative impact on performance.

Table 3.7 summarizes the performance results. Surprisingly, the overall winner is Random Forest, outperforming all five neural models and four existing AA methods. As to per-class F1 scores, Random Forest, a robust non-linear model, accurately solved the AA problem across all nine test sets (one human and eight neural generators). Most generated texts were relatively easy to identify their authorship, giving up high F1 scores (especially the generators such as CTRL, GPT, XLM, XLNET, and PPLM).

The most challenging test set turns out to be both Human and GROVER that yields relatively low average F1 scores across all of classical, neural, and existing AA models (0.5423 and 0.5542, respectively). Also, interestingly, neural classifiers are able to classify FAIR very accurately unlike classical or existing AA models, while classical models, especially Random Forest and SVM, perform better for tough test sets such as GROVER and Human.

3.9 Discussion

3.9.1 P1: Same Method or Not

As expected, we find that the balanced setting yields significantly higher F1 scores across five neural models than the imbalanced setting. However, P1 is still nontrivial to solve, especially in the imbalanced setting, as can be seen in Figure 3.1, where many deepfake-generated texts are shown to be linearly inseparable. Furthermore, from Table 3.3 and Section 3.4, we can see that while some generators generate similar texts, all generated texts still possess distinct qualities that are leveraged in P1, achieving F1=0.9813 in the imbalanced setting. It is harder to grasp these distinct characteristics when looking at a single piece of text. As such, the comparison of two texts in the setting of P1 offers an advantage to the task.

3.9.2 P2: Human vs. Deepfake

We find that RoBERTa-tuned often outperforms neural classifiers in the individual human vs. deepfake setting, except for the case of GROVER (Table 3.5). RoBERTa-tuned outperforms all competing models in distinguishing deepfake texts from human texts, incredibly well on GPT-2 texts (achieving F1=0.9792), probably due to sufficient training on GPT-2 data. Next, we find that GROVER-DETECT underperforms in classifying the other deepfake-generated texts in Table 3.5, but performs well on Human vs. GROVER achieving the F1 score of 0.8135. This is because it was trained to detect GROVER-generated texts. For the collective settings, however, both RoBERTa and Parallel_CNN have similar F1 scores, while outperforming the rest by significant margins.

3.9.3 P3: Authorship Attribution

For this setting, in Table 3.7, we compare different settings, including (1) the use of GloVe word embedding with Embedding, RNN, and CNN-RNN; (2) no word embedding with Parallel_CNN and Stacked_CNN; (3) the use of linguistic features with classical learning algorithms; and (4) n-grams and POS-tags with state-of-the-art AA methods. In this task, we learn that the more accessible generators to classify are CTRL, XLM, and XLNET, while the harder ones are Human, GROVER, FAIR, and GPT-2. This can be seen in Tables 3.5 and 3.3, where the more demanding generators underperform, and score highly in *LIWC-Analytic* and *LIWC-Article*, respectively. This is vice versa for the more accessible generator. We also find that the linguistic features effectively solve P3, slightly better than state-of-the-art AA solutions, and (simple) neural classifiers. The top stylistic features are *word count, article, period, word-per-sentence count, auxiliary verb, preposition, comma*. We expect this result will change in the future when: (1) the quality of deepfake-generated texts improve, losing revealing linguistic cues, and (2) neural models are trained better with an enormous amount of data and more powerful architectures.

One may wonder if some results with high F1 scores to solve P3 in Table 3.7 are simply due to the fact that different generators tend to generate texts on different topics (with non-overlapping word usage, thereby affecting embedding to neural models). In addition, while we only attempt to collect our articles from the domain of "politics," some other domains may have been added unintentionally. However, when we solve P3 using the combination of bigram and trigram models with top-20 LDA-extracted topics, we achieve only 0.38 as the overall average F1 score. Therefore, we believe that simple topical analysis of generated texts cannot solve P3 well.

3.10 Summary

We have conducted comprehensive experiments on three versions of the Authorship Attribution (AA) problem: (1) the same method or not, (2) human vs. deepfake (Turing Test), and (3) who is the author. Notable findings from our empirical evaluation include: (1) not all deepfake text generation methods generate high-quality human-mimicking texts–in particular, GPT-2, GROVER, and FAIR generated better-quality texts and (2) using specific linguistic features and simple neural architectures, we can solve three problems reasonably well, except GPT-2 and FAIR in P2 and GROVER in P3.

Balanced	#train	#valid	#test
P1	68,896	7,656	19,139
P2	2,985	426	853
P3	6,825	881	1,888

Imbalanced	#train	#valid	#test
P1	62,157	6,907	17,266
P2	6,825	881	1,888

Table 3.8: Details on Train, Validation, and Test Set Splits

3.11 Reproducibility

3.11.1 Implementation, Infrastructure, Software, and Data

We run all experiments using either P100 or Titan Xp GPU card on a standard server machine with 16GB of RAM. We utilize deep learning platform *Ludwig* (v.0.2.1) with *Tensorflow* (v.1.15.0) backend to develop and evaluate all text classification models in the paper. For classical ML, we utilize *scikit-learn* (v.0.22.1) library. All implementations are done using *python* language (v.3.0). For generating text, we adopt various models implementation provided by *huggingface*[11], *PPLM*[12], *grover*[13], and *Fairseq*[14] Github repo. To extract LIWC features, we utilize the LIWC2015 software (v.1.6.0)[15].

3.11.2 Data and Preprocessing

We generate all the text following the description in Section 3.3. Since the generated text of some deepfake algorithms includes artificial tokens such as <eos> and <sos>, we remove these tokens from the results. We also ensure that the prompts (i.e., article titles) are appended to every generated article. For P3, we use all the generated text by 9 methods (human and eight deepfake algorithms), resulting in a dataset with balanced label distribution. For P1, creating datasets generators' pairs is a combinatorial problem,

[11]https://github.com/huggingface
[12]https://github.com/uber-research/PPLM
[13]https://github.com/rowanz/grover
[14]https://github.com/pytorch/fairseq
[15]https://liwc.wpengine.com

which will create a very large dataset. Instead, we sample from each possible pairs of generators K samples while maintaining the relative distribution among them, resulting in the imbalanced dataset. Then, we adjust K and under-sample negative samples with 1:8 ratio to create the balanced dataset for P1. For P2, we curated the imbalanced dataset from P3, with 1 human and 8 deepfake generators. Then, we under-sample negative sample with 1:8 ratio to create a balanced dataset for P2. For each task P1, P2, and P3, we then split to train, validation, and test set with 7:1:2 ratio. Table 3.8 summarizes statistics of datasets used for each task in balanced and imbalanced scenario, respectively. Also, using language_check, a python package for detecting and correcting grammatical errors, we found that most generators had less than a 3% grammatical error rate, except for XLM which had a 14% error rate.

3.11.3 Running Time

All experiments take an average running time of around 2 minutes for each training epoch. Depending on the text encoders being utilized, and one training epoch can take as low as 10 seconds (Embedding model) to as long as 8 minutes (CNN-RNN model).

3.11.4 Training and Model's Parameters

See Table 3.9 for both the parameters and hyper-parameters used in training our deep learning models.

3.11.5 Evaluation Metrics

We use standard *Precision (P)*, *Recall (R)*, and *F1* scores as the main evaluation metrics throughout the paper. We first construct a confusion matrix and calculate those scores as follows.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = 2\frac{P \cdot R}{P + R} \tag{3.1}$$

where TP is True Positive, FP is False Positive, FP is False Positive and FN is False Negative predictions.

Parameter	**Value**
Max Words	500
Vocabulary Size	20,000
Early Stop	2
Batch Size	256
Learning Rate	0.01
Adam Optimizer	β_1: 0.9, β_2: 0.999, ϵ: 1e-08
Embedding Size	300
Stacked CNN Kernel Sizes	7, 7, 3, 3, 3 and 3
Stacked CNN Pool Sizes	3, 3, 3, 3, 3, and 3
Parallel CNN Kernel Sizes	2, 3, 4, and 5
RNN Hidden Size	256
FCN Layers (before Softmax)	256 - 256
Dropout	0.5

Table 3.9: Model's Parameters and Training's Hyper-Parameters

Chapter 4 | TuringBench: A Benchmark Environment for Turing Test in the Age of Deepfake Text Generation

4.1 Introduction

To build accurate detectors of deepfake texts, sufficient data is required but lacking. Therefore, we create a benchmark environment, TuringBench, to combat the obvious security issue language models could pose. Just in line with benchmark environments such as SQuAD [167] and GLUE [168] that tremendously facilitate the progress of Natural Language Understanding, we build the first benchmark for Authorship Attribution in the form of the Turing Test by including humans and neural language models.

The TuringBench Environment comprises benchmark datasets, benchmark tasks, and a website to host leaderboards. This benchmark dataset is created by collecting 10K news articles (mostly in politics) written by journalists in media outlets such as CNN, Washington Post, etc. Using the Title of each article, we Prompt 19 selected deepfake text-generators to generate an article similar to the human-written one. This creates 200K articles with 20 labels (or authors). Next, we have two benchmark tasks - *Turing Test* and *Authorship Attribution*. The *Turing Test* task is modeled after the *Turing Test* concept [4], where if a machine shows intelligent behavior or characteristics usually attributed to a human, then the machine has passed the test. In this scenario, the goal

PROMPT: *Some investigators struggle to reach Comey*

HUMAN: Some congressional investigators want to bring former FBI Director James Comey to the Hill to tell them about his meetings with President Donald Trump but they can't find him.

GPT-2: Some investigators struggle to reach Comey and his team because the FBI's headquarters is in Northern Virginia. But the FBI has a lot of other people who can help.

FAIR: We have a duty to be vigilant. James Comey's testimony today should make him think twice about it. The White House said Trump had spoken with Comey during the hour-long meeting and that all the senior U.S. officials were given a "heads up" on the situation.

GPT-3: "Like Louis XVI, he doesn't see the storm growing in the distance," says the Democratic operative. The lack of specifics, even from surrogates on Trump's behalf, forces well-known Democrats to point out the obvious.

Figure 4.1: Snippets of **PROMPT** (a news article title), **HUMAN** vs. three examples of deepfake-generated texts (**GPT-2**, **FAIR** and **GPT-3**).

is to cause the machine to fail the *Turing Test*. Thus, we define this benchmark task as a binary classification problem with *human* and *deepfake* labels. Given 19 deepfake text-generators, there are 19 *Turing Test* subtasks with 19 human-deepfake pairs. See Figure 4.1 for snippets of this dataset.

Furthermore, we understand that due to the ubiquitous nature of these neural language models, distinguishing deepfake-generated texts from human-written ones is no longer sufficient. It is now also important we inquire as to which particular deepfake text-generator authored a piece of text. To this end, the *Authorship Attribution* task aims to assign authorship to one of the many text-generators. We study 20 authors for this task, however, as we have observed, this can easily become 2K authors very soon which will grossly exacerbate the difficulty of this task. Finally, to host all these tasks and datasets, we build a TuringBench website with leaderboards for each benchmark task and call for participation in tackling this very relevant and non-trivial problem.

Lastly, we compare State-of-the-art (SOTA) and baseline *Turing Test* and *Authorship Attribution* models. From the experimental results, we observe that we need more complex models to accurately distinguish deepfake-generated texts from human-written ones, including text-generators that are yet to be created.

Text Generator	**Description**
Human	We collected news titles (mostly Politics) and contents from CNN, Washington Post, and Kaggle. The Kaggle datasets had news articles from 2014–2020, and 2019–2020 for the CNN and Washington Post news articles. Next, we removed articles that did not have the desired word length (i.e., 200–500). This resulted in 130K articles, but only 10K was used for the article generations. See data generation process in Figure 4.3.
GPT-1	Texts are generated with the Hugging Face github repo [13].
GPT-2	We use 5 GPT-2 pre-trained models - **PyTorch** model, **small** (124 million parameters), **medium** (355 million parameters), **large** (774 million parameters), and **x-large** (1558 million parameters) to generate texts.
GPT-3	Texts are generated with the OpenAI GPT-3 API using the *davinci* engine.
GROVER	We use code from repo to generate from Grover's 3 pre-trained models: **GROVER-base**, **GROVER-large**, **GROVER-mega**.
CTRL	Conditional Transformer Language Model For Controllable Generation uses control codes to guide generation. We use *News* control code to generate long articles.
XLM	We generated texts using Hugging Face repo [13].
XLNET	We generated texts with: 2 XLNET pre-trained models: **XLNET-base**, and **XLNET-large** using Hugging Face.
FAIR_wmt	We use two Facebook's FAIR English models - **wmt19** [8] and **wmt20** [9] to generate texts with FAIRSEQ sequence modeling toolkit.
TRANSFORMER_XL	We generated texts with this language model's setup on Hugging Face [13].
PPLM	PPLM fuses GPT-2's pre-trained model with bag of words to generate more specific texts. We used the *Politics* bag of words model to generate texts. Next, we fused PPLM with two pre-trained models (i.e., distilGPT-2, and GPT-2) and generated texts with them, forming: **PPLM_distil**, **PPLM_gpt2**. These models are gotten from the Hugging Face model repository.

Table 4.1: Description of the text generators in the TuringBench dataset.

4.2 The TuringBench Environment

Figure 4.2 overviews the framework of the TuringBench Environment.

4.2.1 Chosen Language Models

We generated texts using 10 language model architectures - *GPT-1* [7], *GPT-2* [38], *GPT-3* [31], *GROVER* [16], *CTRL* [10], *XLM* [39], *XLNET* [40], *FAIR* [8,9], *TRANSFORMER-XL* [41], and *PPLM* [11]. In addition, some of these language models have multiple pre-trained models and thus, we were able to generate texts with 19 neural deepofake text-generators. We choose these 10 language model architectures because they are currently considered as the SOTA text-generators, many of the text-generators on Hugging Face's model repo are variants of these language models, and both their pre-trained models and

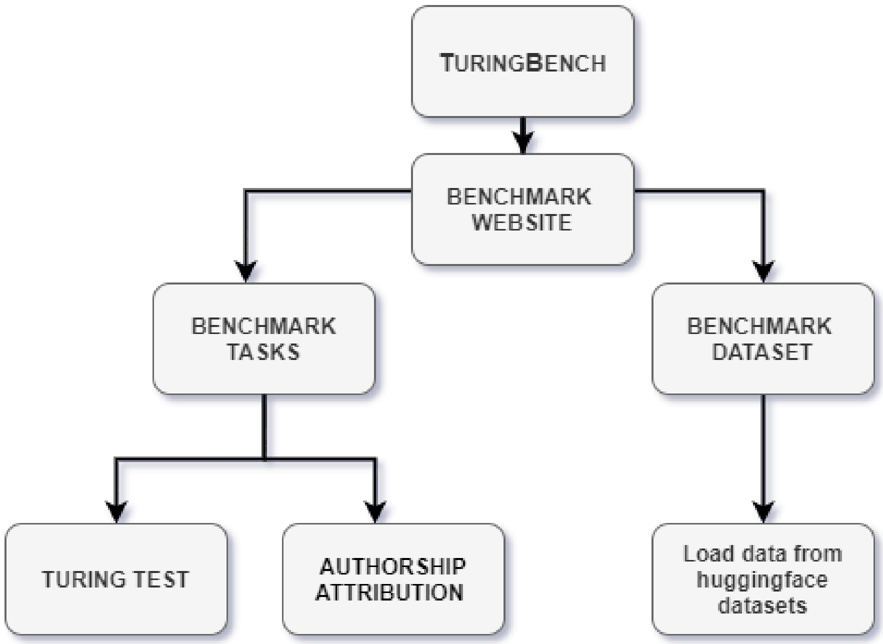


Figure 4.2: The TuringBench Environment.

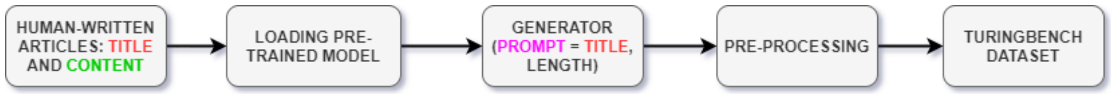

HUMAN-WRITTEN ARTICLES: TITLE AND CONTENT
LOADING PRE-TRAINED MODEL
GENERATOR (PROMPT = TITLE, LENGTH)
PRE-PROCESSING
TURINGBENCH DATASET

Figure 4.3: The TuringBench Data Collection, Generation, and Building process.

codes were publicly available.

To generate texts, all 19 neural generators require a short prompt and a specified number of words to generate texts. Table 4.1 (and Appendix) describes each language model in detail. Figure 4.3 illustrates the data creation process. Table 4.2 summarizes the stats of dataset and the model sizes.

4.2.2 TuringBench Benchmark Tasks

4.2.2.0.1 The Turing Test (TT) Task Our proposed Turing Test task aims to answer the question: *Can we determine if a piece of text is human-written or deepfake-generated?* This task is formulated as a binary classification problem with two labels – *human* and *deepfake* – modeled after the classical *Turing Test* problem. The *Turing Test* examines the ability of a deepfake text-generator to exhibit intelligible behavior ascribed to humans. The goal is to build a model that causes the deepfake-generated texts to fail the *Turing*

Text Generator	# of words (AVG ± Std. Dev.)	# of sentences (AVG ± Std. Dev.)	Model Parameter Size
Human	232.7 ± 42.0	15.0 ± 6.6	N/A
GPT-1	316.7 ± 12.9	10.5 ± 3.7	117M
GPT-2_small	118.6 ± 61.0	4.0 ± 3.8	124M
GPT-2_medium	120.9 ± 66.0	4.2 ± 3.7	355M
GPT-2_large	119.7 ± 62.1	4.1 ± 3.8	774M
GPT-2_xl	117.8 ± 63.3	4.1 ± 3.8	1.5B
GPT-2_PyTorch	178.9 ± 55.4	7.03 ± 4.8	344M
GPT-3	129.5 ± 54.9	5.0 ± 3.7	175B
GROVER_base	299.2 ± 108.6	9.4 ± 6.9	124M
GROVER_large	286.3 ± 101.3	8.7 ± 5.9	355M
GROVER_mega	278.9 ± 97.6	9.2 ± 6.1	1.5B
CTRL	398.1 ± 64.8	20.0 ± 10.6	1.6B
XLM	387.8 ± 30.3	4.2 ± 1.7	550M
XLNET_base	226.1 ± 97.5	11.6 ± 7.9	110M
XLNET_large	415.8 ± 53.2	4.3 ± 2.1	340M
FAIR_wmt19	221.2 ± 66.6	14.6 ± 6.0	656M
FAIR_wmt20	100.6 ± 28.1	5.1 ± 3.0	749M
TRANSFORMER_XL	211.7 ± 53.9	9.8 ± 3.1	257M
PPLM_distil	156.9 ± 40.1	10.7 ± 3.6	82M
PPLM_gpt2	188.9 ± 52.0	11.9 ± 4.5	124M

Table 4.2: Summary statistics of the TuringBench dataset.

Test. Lastly, the TT task contains 19 subtasks with 19 human-deepfake pairs (e.g. GPT-2 XL vs. Human, GROVER_base vs. Human, etc.).

4.2.2.0.2 The Authorship Attribution (AA) Task *Authorship Attribution* is the identification and proper assignment of the author of a piece of text [169]. Our Authorship Attribution task aims to answer the question: *If we determine that an article is human-written or deepfake-generated, can we further determine which neural language model generated all the articles that are said to be deepfake-generated?* This is a multi-class classification problem modeled after the traditional *Authorship Attribution* problem.

4.2.3 TuringBench Benchmark Dataset

We keep $168,612$ articles out of 200K after cleaning the text (see Appendix for data pre-processing
details), and we build the benchmark dataset for each benchmark task - *TT* and *AA*. For the *TT* task, there are 20 labels (i.e., 19 deepfake text-generators and 1 human), thus we can only have 19 pairs of human vs. deepfake. Therefore, we have 19 datasets for the TT task. To increase the difficulty of the *TT* task, we cut each article in the test set in half, using only 50% of the words. For the *AA* task, we have 1 dataset containing all the

```
from datasets import load_dataset
import pandas as pd

# GPT-1 TT task
TT_gpt1 = load_dataset(
      ’turingbench/TuringBench’,
      name=’TT_gpt1’, split=’train’)
TT_gpt1 = pd.DataFrame.from_dict(
                         TT_gpt1)

# AA task
AA = load_dataset(
        ’turingbench/TuringBench’,
        name=’AA’, split=’train’)
AA = pd.DataFrame.from_dict(AA)
```

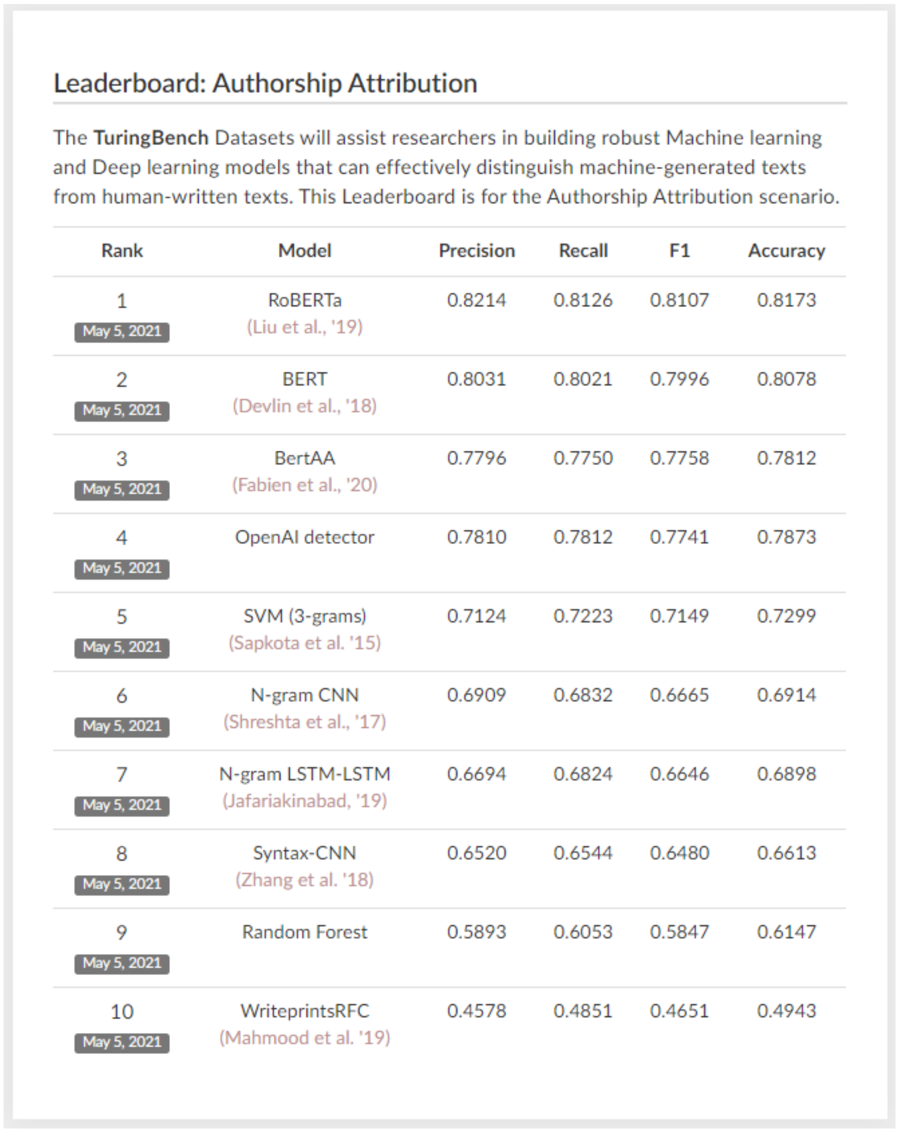
Figure 4.4: Python code for loading the TuringBench datasets using the Hugging Face API.

labels. All datasets have train/validation/test sets which were split using the 70:10:20 ratio, respectively. To avoid topic bias, these sets were carefully split, such that all articles in the sets were unique to each other. Therefore, all articles generated by a prompt belonged only to one set.

To make this dataset public, we added our datasets for each benchmark task and subtask to Hugging Face datasets[1]. Figure 4.4 demonstrates how to load the TuringBench dataset.

4.2.3.0.1 Evaluation Metrics We use the traditional evaluation metrics such as: Precision, Recall, F1 score, and Accuracy to evaluate Machine/Deep Learning models for the benchmark tasks. However, for the TT tasks, we only use F1 scores since it is a more robust measure for the imbalanced datasets.

[1]https://huggingface.co/datasets/turingbench/TuringBench/tree/main



Leaderboard: Authorship Attribution

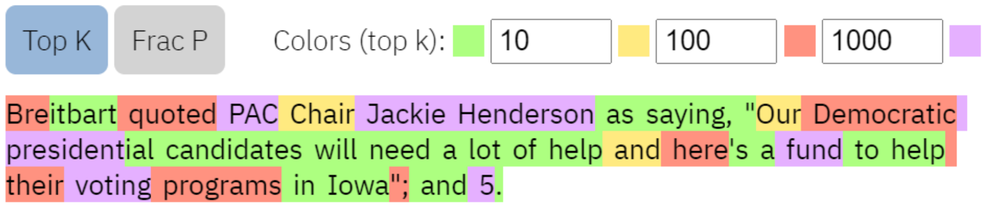
The **TuringBench** Datasets will assist researchers in building robust Machine learning and Deep learning models that can effectively distinguish machine-generated texts from human-written texts. This Leaderboard is for the Authorship Attribution scenario.

Rank	Model	Precision	Recall	F1	Accuracy
1 May 5, 2021	RoBERTa (Liu et al., '19)	0.8214	0.8126	0.8107	0.8173
2 May 5, 2021	BERT (Devlin et al., '18)	0.8031	0.8021	0.7996	0.8078
3 May 5, 2021	BertAA (Fabien et al., '20)	0.7796	0.7750	0.7758	0.7812
4 May 5, 2021	OpenAI detector	0.7810	0.7812	0.7741	0.7873
5 May 5, 2021	SVM (3-grams) (Sapkota et al. '15)	0.7124	0.7223	0.7149	0.7299
6 May 5, 2021	N-gram CNN (Shreshta et al., '17)	0.6909	0.6832	0.6665	0.6914
7 May 5, 2021	N-gram LSTM-LSTM (Jafariakinabad, '19)	0.6694	0.6824	0.6646	0.6898
8 May 5, 2021	Syntax-CNN (Zhang et al. '18)	0.6520	0.6544	0.6480	0.6613
9 May 5, 2021	Random Forest	0.5893	0.6053	0.5847	0.6147
10 May 5, 2021	WriteprintsRFC (Mahmood et al. '19)	0.4578	0.4851	0.4651	0.4943

Figure 4.5: A screenshot of a leaderboard on the TuringBench website.

4.2.4 The Web Environment

To create this TuringBench environment, we built 2 versions of datasets - binary setting (i.e., *human vs. GROVER-large, human vs. GPT-1, etc.*) for the TT tasks, and multi-class setting (i.e., *human vs. GROVER-Large vs. GPT-1 vs. etc.*) for the AA task. To track progress, as shown in Figure 4.5, we create a website where each task and sub-task has its own leaderboard that displays the evaluation metric scores of models. Furthermore, to ensure the integrity of the process, even though contributors can obtain the TuringBench



Breitbart quoted PAC Chair Jackie Henderson as saying, "Our Democratic presidential candidates will need a lot of help and here's a fund to help their voting programs in Iowa"; and 5.

Figure 4.6: Using GLTR [2] on a piece of text generated by GPT-3. Green represents the most probable words; Yellow the 2nd most probable; Red is the least probable; and Purple is the highest improbable words. Deepfake-generated texts are often populated with mostly Green and Yellow words. However, we see that GPT-3-generated texts are very human-like.

TT Model	**Description**
GROVER detector	We use the GROVER-Large discriminator that is trained to detect GROVER-generated texts to predict the test labels.
GPT-2 detector	We use the trained weights of RoBERTa-large fine-tuned on GPT-2 XL outputs to predict the *human* and *deepfake* label of the test dataset.
GLTR	In the GLTR demo, the words are color coded to improve human detection of deepfake-generated texts. Top 0-10 probable words are green; top 10-100 probable words are yellow; top 100-1000 probable words are red and top greater than 1000 words are purple. See Figure 4.6 for an example of using GLTR and interpretation of its color schemes. Thus, we define human-written texts to be any article that 10% or more of the words belong in the top >1000 (i.e., purple words).
BERT	We fine-tune *bert-base-cased* on the train set and classify on the test set.
RoBERTa	We fine-tune RoBERTa-base, a variant of BERT with the train set.

Table 4.3: Description of the Turing Test (TT) models.

datasets from Hugging Face datasets, we still ask contributors to submit their code and/or trained model weights for private testing. After testing, we update the website with the new models' scores. Lastly, we rank the model performance using the F1 score from best to worst.

4.3 Experiments

We experiment with several SOTA and baseline models as summarized in Table 4.3 for **Turing Test** and Table 4.4 for **Authorship Attribution**, and Table 4.5 and Table 4.6 show their results.

AA Model	**Description**
Random Forest	Using TF-IDF to represent the data, we classify the texts with Random Forest.
SVM (3-grams)	We represent the texts as 3-grams and classify the texts with SVM.
WriteprintsRFC	Writeprints features + Random Forest Classifier.
OpenAI detector	We re-purposed RoBERTa-base (*roberta-base-openai-detector*) model that was originally fine-tuned on GPT-2 XL outputs to detect deepfake texts, by training the model as a multi-classifier for the AA task.
Syntax-CNN	Use Part-Of-Speech to capture the syntax of the texts and classify the texts with CNN
N-gram CNN	Represent the data with n-grams (uni-grams) and classify texts with CNN
N-gram LSTM-LSTM	Represent the data with n-grams (uni-grams) and classify texts with LSTM
BertAA	Using BERT + Style + Hybrid features to achieve automatic authorship attribution. Style features include: *length of text, number of words, average length of words,* etc. and Hybrid features include: *frequency of the 100 most frequent character-level bi-grams* and *the 100 most frequent character-level tri-grams.*
BERT-Multinomial	Using BERT for multi-class classification
RoBERTa-Multinomial	Using RoBERTa for multi-class classification

Table 4.4: Description of the Authorship Attribution (AA) models.

Human vs.	**Human Test (deepfake)**	**Human Test (human vs. deepfake)**	**GROVER detector**	**GPT-2 detector**	**GLTR**	**BERT**	**RoBERTa**	AVG
GPT-1	0.4000	0.5600	0.5792	0.9854	0.4743	0.9503	0.9783	0.7935
GPT-2_small	0.6200	0.4400	0.5685	0.5595	0.5083	0.7517	0.7104	0.6197
GPT-2_medium	0.5800	0.4800	0.5562	0.4652	0.4879	0.6491	0.7542	0.5825
GPT-2_large	0.7400	0.4400	0.5497	0.4507	0.4582	0.7291	0.7944	0.5964
GPT-2_xl	0.6000	0.4800	0.5549	0.4209	0.4501	0.7854	0.7842	0.5991
GPT-2_PyTorch	0.5000	0.5600	0.5679	0.5096	0.7183	0.9875	0.8444	0.7255
GPT-3	0.4400	0.5800	0.5746	0.5293	0.3476	0.7944	0.5209	<u>0.5534</u>
GROVER_base	0.3200	0.4200	0.5766	0.8400	0.3854	0.9831	0.9870	0.7544
GROVER_large	0.4800	0.5800	0.5442	0.5974	0.4090	0.9837	0.9875	0.7044
GROVER_mega	0.5400	0.4800	0.5138	0.4190	0.4203	0.9677	0.9416	0.6525
CTRL	0.5000	0.6900	0.4865	0.3830	0.8798	0.9960	0.9950	0.7481
XLM	0.6600	0.7000	0.5037	0.5100	0.8907	0.9997	0.5848	0.6978
XLNET_base	0.5200	0.5400	0.5813	0.7549	0.7541	0.9935	0.7941	0.7756
XLNET_large	0.5200	0.5200	0.5778	0.8952	0.8763	0.9997	0.9959	0.8690
FAIR_wmt19	0.5600	0.5600	0.5569	0.4616	0.5628	0.9329	0.8434	0.6715
FAIR_wmt20	0.5800	0.2800	0.5790	0.4775	0.4907	0.4701	0.4531	**0.4941**
TRANSFORMER_XL	0.5000	0.5000	0.5830	0.9234	0.3524	0.9721	0.9640	0.7590
PPLM_distil	0.5600	0.4400	0.5878	0.7178	0.6425	0.8828	0.8978	0.7457
PPLM_gpt2	0.5600	0.5000	0.5815	0.5602	0.6842	0.8890	0.9015	0.7233
AVG	0.5358	0.5132	0.5591	0.6032	0.5681	**0.8799**	<u>0.8280</u>	

Table 4.5: Compared Human Test vs. Test F1 scores of Turing Test models (**bold** and <u>underlined</u> are #1 and #2 performance, respectively). Human Test (deepfake) asked humans to decide if a given article is deepfake or not, while Human Test (human vs. deepfake) asked humans which of the two given texts is deepfake-generated.

AA Model	**P**	**R**	**F1**	**Accuracy**
Random Forest	0.5893	0.6053	0.5847	0.6147
SVM (3-grams)	0.7124	0.7223	0.7149	0.7299
WriteprintsRFC	0.4578	0.4851	0.4651	0.4943
OpenAI detector	0.7810	0.7812	0.7741	0.7873
Syntax-CNN	0.6520	0.6544	0.6480	0.6613
N-gram CNN	0.6909	0.6832	0.6665	0.6914
N-gram LSTM-LSTM	0.6694	0.6824	0.6646	0.6898
BertAA	0.7796	0.7750	0.7758	0.7812
BERT-Multinomial	<u>0.8031</u>	<u>0.8021</u>	<u>0.7996</u>	<u>0.8078</u>
RoBERTa-Multinomial	**0.8214**	**0.8126**	**0.8107**	**0.8173**

Table 4.6: Performance of Authorship Attribution models (**bold** and <u>underlined</u> are #1 and #2 performance, respectively).

4.3.1 Results from Turing Test

The *Turing Test* task is formulated as a binary classification problem with *human* and *deepfake* labels. In order to make the TT task even more difficult, we train and validate on the full articles generated by the text-generators and test on only 50% of the words of each article in the test set. We intend to capture the differences that will exist between train and test data in the real world in this scenario.

We compare 3 SOTA TT models - GROVER detector [16], GPT-2 detector [170], and GLTR [2]. We observe in Table 4.5 that the average F1 scores are 0.56, 0.60, and 0.57, respectively. Next, using other text classifiers such as BERT [82] and RoBERTa [110] brings a significant improvement in F1 scores (0.85 for both BERT and RoBERTa).

This performance improvement occurs mainly because BERT and RoBERTa are fine-tuned with the train set of each TT subtasks, while the TT models' pre-trained models were used to classify the test set without any further training.

Additionally, averaging over all the 5 TT models, we find that FAIR_wmt20 and GPT-3, the most recent text-generators in the list, achieve the lowest average F1 score (0.49 and 0.55), thus making them the language models that produce the most indistinguishable texts, while XLNET_large has the highest average F1 score (0.87) using all TT models. XLNET has a high F1 score because it implements a text padding technique for generation which often negatively affects the generation quality.

We also run two human experiments using the Amazon Mechanical Turk (AMT) environment, recruiting workers with at least 95% approval rate of Human Intelligence Task (HIT). In the experiments, we randomly sampled 50 articles per each language model (across all 19 models) and performed two tests, where workers (1) vote if a given article is deepfake or not, and (2) vote which of two given articles is deepfake-generated. These experiments yielded the AVG-accuracies of 0.535 and 0.513 (random-guess=0.5), respectively.

This part of experiments was reviewed and approved by the Institutional Review Board of our institution.

4.3.2 Results from Authorship Attribution

Since there are 20 labels in AA, the chance performance is at 0.05 (i.e., 5% in accuracy). Due to this difficulty, we use the full article contents in the test set. We compare different SOTA and popular techniques for automatic authorship attribution for our AA task including Random Forest, SVM (3-grams) [171], WriteprintsRFC [100], OpenAI detector[2], Syntax-CNN [83], N-gram CNN [84], N-gram LSTM-LSTM [85], BertAA [172], BERT-Multinomial [82], RoBERTa-Multinomial [110]. We find that BERT and RoBERTa outperform all the AA models, sometimes significantly, achieving the F1 scores of 0.80 and 0.81, respectively.

Interestingly, we observe that OpenAI detector, a RoBERTa-base model fine-tuned on GPT-2 XL outputs, does not outperform BERT-Multinomial and RoBERTa-Multinomial for this AA task although it performs comparatively, achieving a 0.77 as F1 score. BertAA achieves a slightly better F1 score (0.78).

4.4 Discussion

We present several observations from our experimental results.

1. **Both TT and AA tasks are non-trivial:** The average F1 score for each human vs. deepfake subtask and TT model is below 0.87, with FAIR_wmt20 achieving the lowest (0.49). FAIR_wmt20 is the newest text-generator in our list and before that we have GPT-3 which achieves the second lowest average F1 score (0.55). This suggests a trend that as newer text-generators get built, generated texts will become even more human-like, making the TT and AA tasks more difficult.

[2]https://huggingface.co/roberta-base-openai-detector

Additionally, the difficulty of the AA task is further demonstrated by the PCA plot of linguistic features LIWC of the TuringBench dataset in Figure 4.7. Using LIWC to capture stylistic signatures of authors has been studied [26, 173]. However, we observe that there are quite a few overlaps in linguistic features across different authors (i.e., language models). This makes these authors' writing styles linearly inseparable.

2. **No one size fits all:** We observe in Table 4.5 that there is no one detection model that performs well across all 20 TT tasks. For instance, while BERT achieved the highest average F1 score, it still underperformed in detecting FAIR_wmt20. However, GROVER detector achieved a highest F1 score in detecting FAIR_wmt20.

3. **Humans detect deepfake texts at chance level:** First two columns of Table 5 show the results of human detection test. In the first AMT-based tests, we randomly sampled 50 deepfake texts and asked humans to decide if the given text is human-written or deepfake-generated (i.e., humans do not know whether they are shown only deepfake texts in the test). In the second test, we showed two texts at random, one written by humans and the other deepfake-generated, and asked humans to decide which of the two is deepfake-generated (i.e., humans know that at least one of two is deepfake-generated).

   Based on the average accuracies of two human tests, by and large, we observe that humans currently differentiate deepfake texts from human-written ones, not much better (i.e., 0.535 and 0.513) than the level of random guessing (i.e., 0.5).

4. **Not all text-generators are created equal:** As shown in Table 4.5, the average F1 score for each human vs. deepfake subtask and TT model is below 0.87, with FAIR_wmt20 achieving the lowest (0.49). Consequently, this suggests that FAIR_wmt20 is the most sophisticated text-generator and thus, the hardest to detect. Other generators that are also hard to detect based on their < 0.62 F1 scores are: GPT-3, GPT-2_small, GPT-2_medium, GPT-2_large, and GPT-2_XL.

5. **Sophisticated deepfake-generated texts often get detected as human-written:** We observe an interesting phenomenon with these SOTA TT models. For instance, even though the labels in the binary classification task are approximately evenly split, GPT-2 detector and GLTR achieve below F1 score of 0.4 in some subtasks. This happens because TT models do not generalize well to those specific text-generators (i.e.,

GROVER_base, CTRL, GPT-3, TRANSFORMER_XL) and mistakenly predicts the majority of the texts as *human-written*.

6. **TT models do not always perform as expected:** While both GROVER and GPT-2 detectors are trained to detect GROVER-generated and GPT-2-generated texts, respectively, they underperform in detecting those texts. For instance, GROVER detector performs the best in detecting PPLM_distil and PPLM_gpt2 texts, while GPT-2 detector performs significantly better at detecting GPT-1, TRANSFORMER_XL and XLNET_large texts.

7. **Length of texts does not affect model performance:** Due to the varying length of texts (i.e. 100-400) in Table 4.2, we plot the length of generated texts vs. the F1 scores of TT models in Figure 4.8. However, the figure suggests that there is no clear correlation between model performance and length of texts for all models except RoBERTa. This suggests that RoBERTa performance is text length-dependent.

8. **Traditional AA models cannot fully capture an author's style "yet":** SOTA AA models cannot capture all of the stylistic features of human and deepfake text-generators. From Figure 4.7 we observe that the psycho-linguistic features of the 20 authors in the TuringBench dataset are too similar, causing them to overlap in the plot. This suggests that deepfake texts are becoming more similar to human-written texts in styles.

   Therefore, traditional ways to capture an author's writing style will no longer be sufficient to achieve accurate automatic authorship attribution. This is further confirmed in the performance of classical AA models such as SVM and Random Forest. Similarly, we find that even deep learning-based AA models are still unable to fully capture the distinct writing styles of all 20 authors. These results suggest that one needs to develop a model that can unearth more subtle yet distinct patterns that exist across 20 models.

9. **Humans have a wide writing style range:** In Figure 4.7, we observe that human-written features spread out all over the plot, while all deepfake-generated texts stay in little pockets of the plots. This suggests that humans may have a wider range of writing levels/styles, while deepfakes have a more limited range of writing levels/styles (e.g., high school to college).

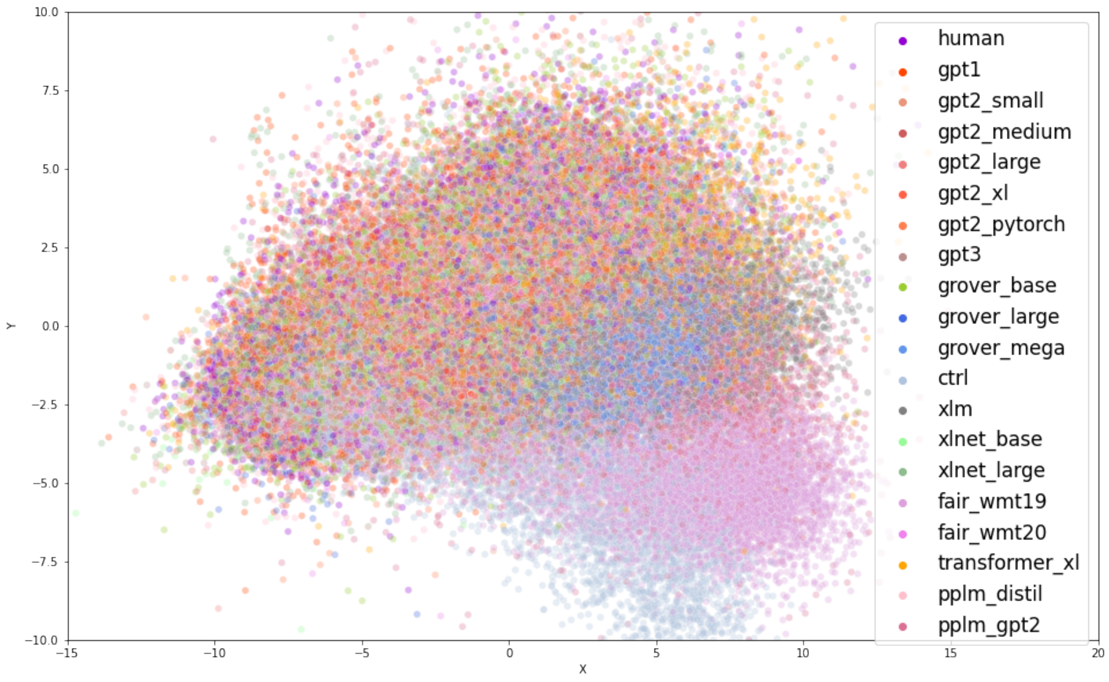


Figure 4.7: PCA plot of psycho-linguistics features of the TuringBench dataset, using LIWC to attempt to capture the stylistic signatures of the dataset

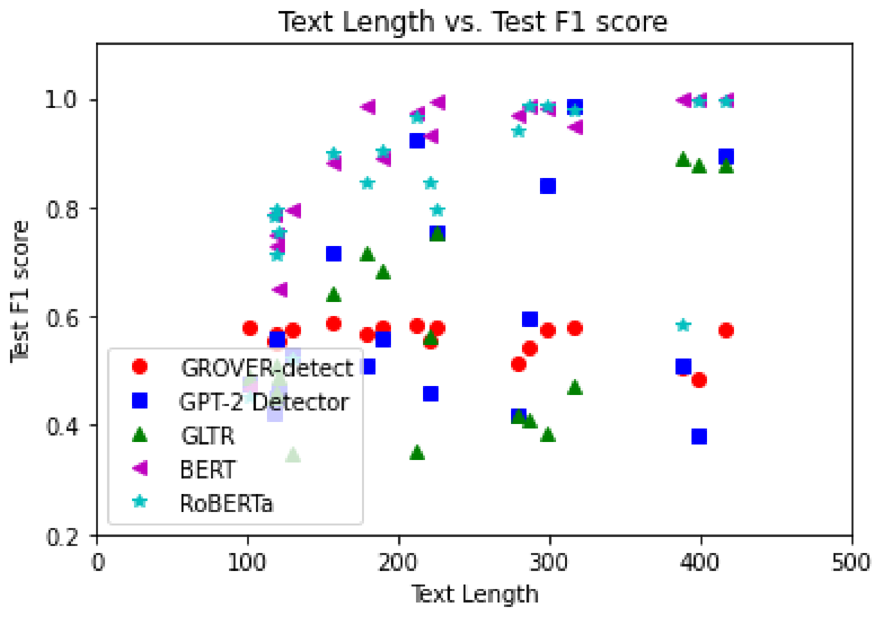


Figure 4.8: Despite the varying lengths of the generated texts (100–400) in Table 4.2, no correlation between text length and F1 score was found.

4.5 Summary

In this paper, we have introduced the TuringBench environment and its preliminary results for both Turing Test (TT) and Authorship Attribution (AA) tasks. While varied, overall, (1) many contemporary language models can generate texts whose qualities are, to human eyes, indistinguishable from human-written texts, and (2) while some computational solutions for both TT and AA tasks can differentiate human-written texts from deepfake-generated ones much better than random guessing, overall, the community needs to research and develop better solutions for mission-critical applications. We hope that the TuringBench environment will provide a platform on which insights into ways to tackle this urgent issue can be developed and shared.

4.6 Reproducibility: Data Generation & Experiments

4.6.1 Data Generation Implementation

Generating texts with these Language models is very computationally expensive. Some of the python code used to generate the texts were not written for large-scale generation, so we had to re-purpose it for our task. We mostly used Google Colab pro's GPU - 12GB NVIDIA Tesla K80 to generate our texts. However, since *PPLM* was the heaviest language model computationally, we used a machine with more GPUs - NVIDIA Tesla K80s and P100s.

Most generators took 24 – 72 hours to generate 10K articles. However, *PPLM* took about 430 hours for *PPLM_distil* and about 600 hours for *PPLM_gpt2*. It is important to note that probably a few coding choices could reduce the computational cost of running *PPLM*, we just did not get to it. See the description of building the human dataset and 10 language model architectures used to generate the rest of the dataset. The table also contains the links to the dataset and github repo of some of the models.

4.6.2 Data Pre-processing

Some of the generated texts contain non-English tokens such as $\langle UNK\rangle$, $\langle eos\rangle$, $\langle eod\rangle$, $\langle eop\rangle$, $\langle|endoftext|\rangle$, etc. which we removed. Also, in an attempt to generate texts with the specified word count (i.e., 400), some of the generators had a tendency to

Text Generator	**# of Data samples**
Human	8,854
GPT-1	8,309
GPT-2_small	8,164
GPT-2_medium	8,164
GPT-2_large	8,164
GPT-2_xl	8,309
GPT-2_PyTorch	8,854
GPT-3	8,164
GROVER_base	8,854
GROVER_large	8,164
GROVER_mega	8,164
CTRL	8,121
XLM	8,852
XLNET_base	8,854
XLNET_large	8,134
FAIR_wmt19	8,164
FAIR_wmt20	8,309
TRANSFORMER_XL	8,306
PPLM_distil	8,854
PPLM_gpt2	8,854

Table 4.7: # of data samples in the TuringBench dataset

repeat a particular word multiple times consecutively. This introduced bias into our Machine Learning models, making it easier to detect such generated texts. Therefore, we removed words that were repeated consecutively, leaving only one. Next, those same text-generators also had a tendency to generate texts where a random word would have the last character repeated multiple times. For instance, a word like “expressed", could be spelled like “expresseddddddddddddddddddddddddddddd”. This also made such generators easy to detect, so we removed words of more than 20 characters to get rid of such words. Lastly, the word “CNN” was used heavily by a few generators, making it easier to detect such generators. Therefore, we removed the word, “CNN" from all the articles.

Before pre-processing of the data, we had 200K, and after the process, we have $168,612$. See data distribution in Table 4.7 of the cleaned dataset. We can observe that the distribution of the dataset is still approximately the same. See the detailed description of the 10 unique architectures used to generate texts for the TuringBench dataset, including the links to their code bases in Table 4.8.

[3]https://www.kaggle.com/snapcrack/all-the-news,

TEXT-GENERATORS	DESCRIPTION
Human	We collected news titles (mostly Politics) and contents from CNN, Washington Post, and Kaggle [3] [4] [5] [6] . Next, we removed articles that did not have the desired word length (i.e., 200–500). This resulted in 130K articles, but only 10K was used for the article generations.
GPT-1	Texts are generated with the huggingface github repo[7].
GPT-2	We use 4 GPT-2 pre-trained models - PyTorch model [8], small (124 million parameters), medium (355 million parameters), large (774 million parameters), and extra-large (1558 million parameters) [9] to generate texts.
GPT-3	Texts are generated with the OpenAI GPT-3 API using the *davinci* engine.
GROVER	We use code from repo[10] to generate from Grover's 3 pre-trained models: **GROVER-base**, **GROVER-large**, **GROVER-mega**.
CTRL	Conditional Transformer Language Model For Controllable Generation [11] uses control codes to guide generation. We use *News* control code to generate long articles.
XLM	We generated texts using huggingface repo.
XLNET	We generated texts with 2 XLNET pre-trained models: **XLNET-base**, and **XLNET-large** using huggingface.
FAIR_wmt	We use two Facebook's FAIR English models - **wmt19**[12] and **wmt20**[13] to generate texts with FAIRSEQ sequence modeling toolkit.
TRANSFORMER_XL	We generated texts with this language model's setup on huggingface.
PPLM	PPLM fuses GPT-2's pre-trained model with bag of words to generate more specific texts. We used the *Politics* bag of words model to generate texts', using the code[14], and used the perturbed version. Next, we fused PPLM with two pre-trained models (i.e., distilGPT-2, and GPT-2) and generated texts with them, forming: **PPLM_distil**, **PPLM_gpt2**. These models are gotten from the huggingface model repository[15].

Table 4.8: Description of the Text-generators in the TuringBench dataset.

[4]https://www.kaggle.com/sunnysai12345/news-summary

[5]https://www.kaggle.com/ryanxjhan/cbc-news-coronavirus-articles-march-26

[6]https://www.kaggle.com/patjob/articlescrape

[7]https://github.com/huggingface/transformers

[8]https://github.com/graykode/gpt-2-Pytorch

[9]https://github.com/minimaxir/aitextgen

[10]https://github.com/rowanz/grover

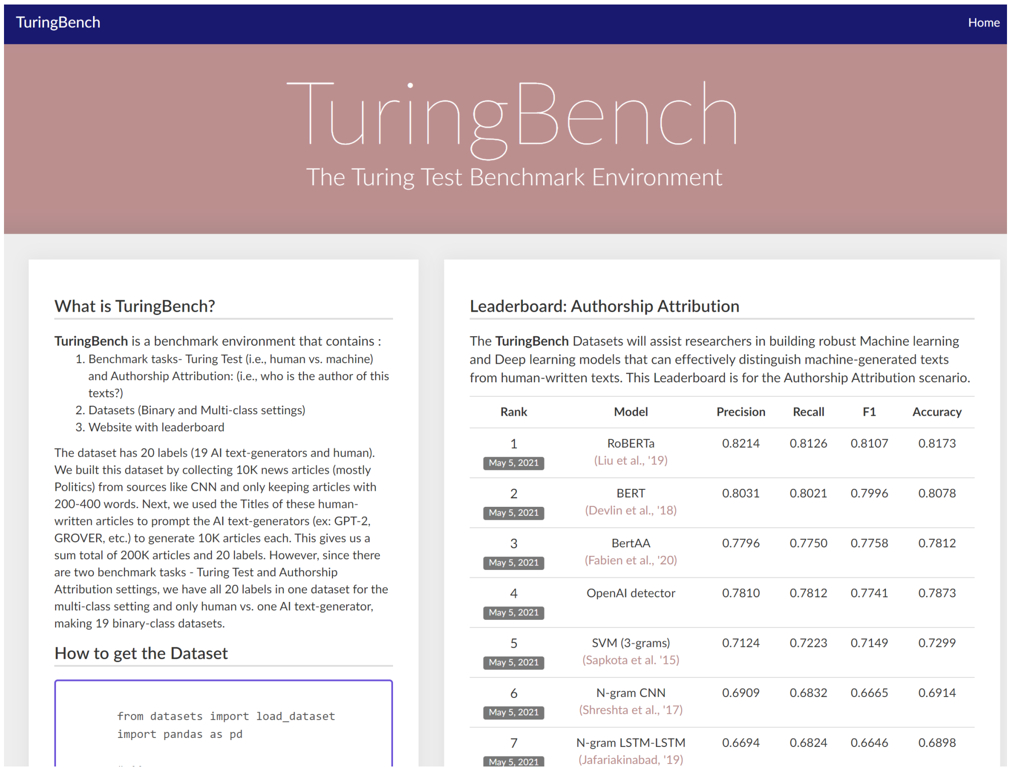
[11]https://github.com/salesforce/ctrl

[12]https://github.com/pytorch/fairseq/tree
/master/examples/wmt19

[13]https://github.com/pytorch/fairseq/tree
/master/examples/wmt20

[14]https://github.com/uber-research/PPLM

[15]https://huggingface.co/models


TuringBench Home

TuringBench

The Turing Test Benchmark Environment

What is TuringBench?

TuringBench is a benchmark environment that contains :

1. Benchmark tasks- Turing Test (i.e., human vs. machine) and Authorship Attribution: (i.e., who is the author of this texts?)
2. Datasets (Binary and Multi-class settings)
3. Website with leaderboard

The dataset has 20 labels (19 AI text-generators and human). We built this dataset by collecting 10K news articles (mostly Politics) from sources like CNN and only keeping articles with 200-400 words. Next, we used the Titles of these human-written articles to prompt the AI text-generators (ex: GPT-2, GROVER, etc.) to generate 10K articles each. This gives us a sum total of 200K articles and 20 labels. However, since there are two benchmark tasks - Turing Test and Authorship Attribution settings, we have all 20 labels in one dataset for the multi-class setting and only human vs. one AI text-generator, making 19 binary-class datasets.

How to get the Dataset

```
from datasets import load_dataset
import pandas as pd
```

Leaderboard: Authorship Attribution

The **TuringBench** Datasets will assist researchers in building robust Machine learning and Deep learning models that can effectively distinguish machine-generated texts from human-written texts. This Leaderboard is for the Authorship Attribution scenario.

Rank	Model	Precision	Recall	F1	Accuracy
1 May 5, 2021	RoBERTa (Liu et al., '19)	0.8214	0.8126	0.8107	0.8173
2 May 5, 2021	BERT (Devlin et al., '18)	0.8031	0.8021	0.7996	0.8078
3 May 5, 2021	BertAA (Fabien et al., '20)	0.7796	0.7750	0.7758	0.7812
4 May 5, 2021	OpenAI detector	0.7810	0.7812	0.7741	0.7873
5 May 5, 2021	SVM (3-grams) (Sapkota et al. '15)	0.7124	0.7223	0.7149	0.7299
6 May 5, 2021	N-gram CNN (Shreshta et al., '17)	0.6909	0.6832	0.6665	0.6914
7 May 5, 2021	N-gram LSTM-LSTM (Jafariakinabad, '19)	0.6694	0.6824	0.6646	0.6898

Figure 4.9: TuringBench website interface

Model	Run-time
GROVER detector	25 – 30 minutes
GPT-2 detector	5 – 10 minutes
GLTR	4 – 5 hours
BERT	25 – 40 minutes
RoBERTa	45 – 1 hour

Table 4.9: TT model Run-time per task

4.6.3 TuringBench Website

We create the TuringBench website using the SQuAD website framework. The website contains a description of the benchmark datasets and benchmark tasks. Each benchmark task has a leaderboard that shows the models used to solve the tasks. These models are rated from best to worst. For the AA tasks, we use the standard Machine learning evaluation metrics such as: *Precision*, *Recall*, *F1 score*, and *Accuracy*. And we use only *F1 score* for the TT task because it is a binary classification problem and *F1 score* is sufficient for the problem. See website interface in Figure 4.9.

4.6.4 Experiments

All experiments, except *GLTR* and *GPT-2 detector* were done using the Google Colab pro's GPU stated above. Experiments with *GLTR* and *GPT-2 detector* were done in a machine with 4 GPUs - NVIDIA Quadro RTX 8000.

4.6.4.1 TT models

Each of the models used its default hyperparameters. There was no hyperparameter tuning performed. We used GROVER-Large discriminator for *GROVER detector*, the weights of Roberta-large fine-tuned on GPT-2 XL outputs for *GPT-2 detector*, and GPT-2 117M model for *GLTR*. None of these models were trained on our dataset. We tested their performance by predicting our test set. Next, we fine-tuned BERT and RoBERTa on our train set and validate these models on our validation set for each TT task. BERT was fine-tuned for 3 epochs and RoBERTa, 3–5 epochs with $2e^{-5}$ learning rate. See Table 4.9 for the run-time of the models.

4.6.4.2 AA models

We used the default hyperparamters of the AA models for the AA task. Also, we did not perform any hyperparameter tuning on these models. *Random Forest* and *SVM* take about 30 minutes – 1 hour to converge. *WriteprintsRFC* took about 15 minutes to converge. *Syntax-CNN*, *N-gram CNN*, and *N-gram LSTM-LSTM* took about 30 minutes to converge. *OpenAI detector* took about an hour to converge. *BERT-Multinomial* and *RoBERTa-Multinomial* took about 1 – 2 hours to converge. *BertAA* took about 5 hours to converge.

Chapter 5 | TopRoBERTa: Topology-Aware Authorship Attribution of Deepfake Texts

5.1 Introduction

We define this problem of *distinguishing deepfake texts from human texts* as *Authorship Attribution* for deepfake texts [174]. Authorship Attribution (AA) is the process of assigning a document to its true author. For deepfake text detection, AA can be divided into two problems - (1) Binary setting: human vs. deepfake; and (2) Multi-class setting:

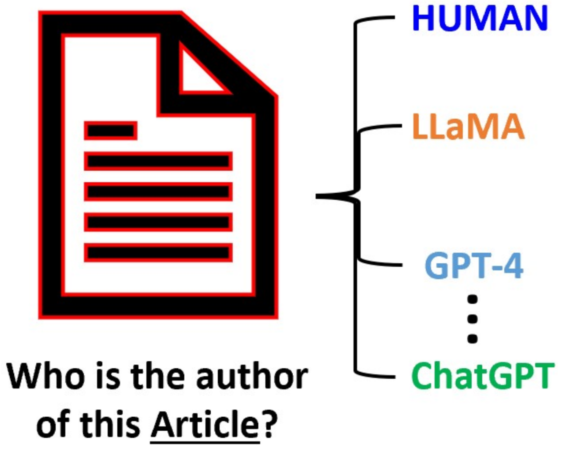


Figure 5.1: Illustration of the Authorship Attribution (AA) problem with multiple authors, including human and many LLM authors

human vs. deepfake-author_1 vs. deepfake-author_2 vs. ... deepfake-author_n. See Figure 5.1 for an illustration of the multi-class setting. The binary setting has been well-studied and is known as the *Turing Test* problem [1], however, the multi-class setting has not been as rigorously studied due to its non-trivial nature. Thus, with the popular usage of ChatGPT (powered by GPT-3.5 or GPT-4), and other LLMs, it will no longer be sufficient to just ask the question - *is this written by human or AI/deepfake algorithms?* but now we know it is AI-generated, can we determine which LLM generated the texts? This can help researchers and policymakers know which models are more likely to be used maliciously and in what context (political propaganda, terrorism recruitment, etc.). Furthermore, if we find that a particular LLM, tends to be used more maliciously in the social media domain, for instance, then we can build a targeted binary detector for this particular LLM.

To distinguish deepfake texts from human-written texts, researchers have proposed several solutions, both utilizing supervised and unsupervised machine learning. Uchendu et al. [174] surveys these solutions well, creating taxonomies that are discussed in Chapter 2. In the supervised learning setting, researchers have developed *stylometric*, *deep learning*, and *hybrid* (ensemble of 2 or more) solutions for deepfake text detection. And for the unsupervised learning setting, only *statistical* solutions have been developed [174]. Intuitively, deep learning- and hybrid-based techniques achieve the best performance in terms of accuracy. However, in terms of adversarial robustness, statistical-based techniques are the most robust models, with hybrid models taking second/first place in adversarial robustness [174]. To that end, we propose a hybrid solution which is an ensemble of statistical and deep learning-based techniques to get both benefits - good performance and robustness. We hypothesize that if our model has adversarial robustness properties, it could also be noise-resistant and thus be robust to out-of-distribution and imbalanced datasets, which could be considered softer adversarial perturbations.

Thus, we propose - **TopRoBERTa**, an ensemble of RoBERTa [110] and Topological Data Analysis (TDA) techniques. We show 2 techniques in which TDA features can be extracted from RoBERTa. First, RoBERTa is used as the base model because it still remains the SOTA for extracting features from text and also because it has over 20K more tokens in its vocabulary than BERT. We apply TDA techniques to the task of deepfake text detection because it is able to capture the true shape of data, in spite of noise [175–178]. To achieve accurate deepfake text detection, we need sufficient data, however due to the expense and restrictive access issues with SOTA LLMs, it is difficult to get sufficiently sized datasets in this field. Consequently, most datasets that exist are grossly imbalanced because they reflect the real world, where there are more human-written texts than deepfake texts. These issues tend to make deepfake text datasets noisy, making TDA a suitable application for deepfake text detection. Thus, we show the robustness of our TDA-based models on imbalanced and noisy datasets - TuringBench [1] & SynSciPass [63]. To build **TopRoBERTa**, we used a 2D version of the $pooled\_output$ to extract TDA features for **TopRoBERTa_pool** and attention weights for **TopRoBERTa_attn**. For both models, we concatenate the regularized $pooled\_output$ and the TDA features and use this new vector as features for classification. **TopRoBERTa_attn** technique is inspired by [24, 179] who use the directed and undirected graphs of the attention weights. We show that **TopRoBERTa_pool** is the superior TDA-based technique as it is more stable, less computationally expensive, and consistently outperforms vanilla RoBERTa.

Finally, with the ensemble of RoBERTa and TDA features we capture syntactic [180, 181], semantic [180, 181], and structural [176] linguistic features with **TopRoBERTa** bringing us closer to accurate authorship attribution of human vs. deepfake texts.

5.2 Related Work: TDA applications in NLP

Topological Data Analysis (TDA) is a technique used to quantify shape and structure in data. Due to this unique ability to obtain the true shape of data, in spite of the noise, it has been implemented in machine learning problems. More recently, the NLP field has seen a recent uptake in TDA applications due to its benefits. TDA has been previously applied to detecting children and adolescent writing [182], law documents analysis [183], movie genre analysis [184], and explanation of syntactic structures of different language families [175, 176]. More recently, TDA techniques have been applied to the deepfake text detection problem [24]. However, they collect the statistical summaries of the TDA representations of BERT attention weights, represented as a directed and undirected graph. Using these representations, they classify deepfake texts with Logistic regression for the binary task - human vs. deepfake. Therefore, for our technique, we train an end-to-end transformer-based model - BERT & RoBERTa with a TDA layer using the BERT or RoBERTa representations as the fine-tuning process continues. Next, [179] uses a similar technique as [24] to show that TDA can improve the robustness of BERT. Finally, TDA has also been applied to representing documents as story trees [185], detecting contradictions in texts [186], examining the linguistic acceptability judgments of texts [187], and finding loops in logic [188].

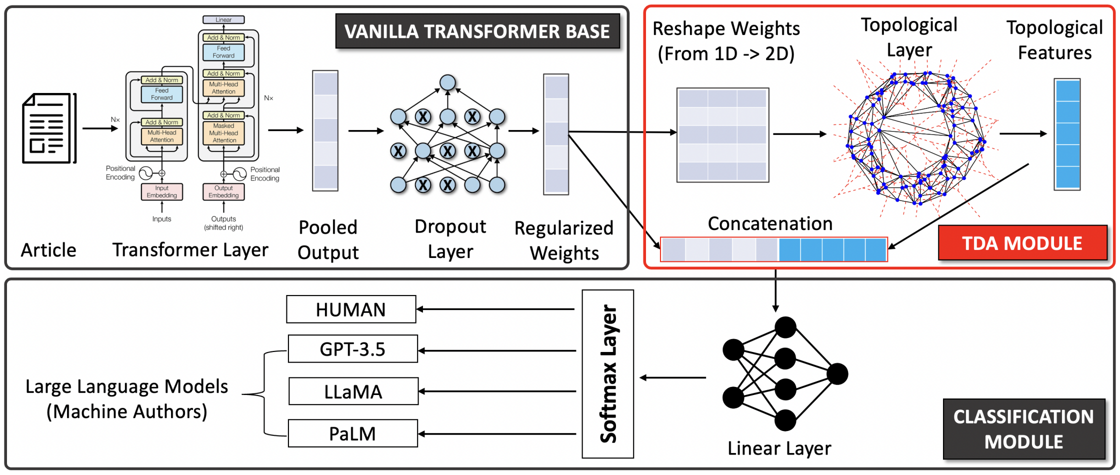


Figure 5.2: Flowchart of the Topological classification algorithm. The **Red** frame indicates our methodology and technique to transform a Vanilla Transformer-based model to a Topological Transformer-based model.

5.3 Topological Data Analysis (TDA) features

Topology is defined as “the study of geometric properties and spatial relations unaffected by the continuous change of shape or size of figures,” according to the Oxford Dictionary. Topological Data Analysis (TDA) is a “collection of powerful tools that have the ability to quantify shape and structure in data”[1]. There are two main TDA techniques - persistent homology and mapper. We will only focus on persistent homology.

TDA is concerned with the formation and deformation of holes. To obtain the true shape of an object, holes are counted in infinite dimensions. However, at about the 3-Dimensions, the features become harder to explain [175]. The dimension of the holes are captured with the *betti numbers* (β_d, d-dimension). The holes in 0-Dimension (β_0) are called connected components and in the 1-Dimension (β_1) and 2-Dimension (β_2), are called loops/tunnels and voids, respectively. The most popular technique to obtain the formation and deformation of these holes is *persistent homology*. Persistent homology is a TDA technique used to find topological patterns of the data [188]. This technique takes in the data and represents it as a point cloud, such that each point is enclosed by a circle. Each circle contains only one point. For this analysis, the aim is to extract the persistent

[1]https://www.indicative.com/resource/topological-data-analysis/

features of the data. Next, it creates simplicial complexes by using different radii within a predefined range to increase the size of the circles. The rule here is due to the increase in the size of the circles, they will begin to overlap each other, therefore, if two circles touch, you draw a line between the two points. This line is called 1-simplex. And if 3 circles touch (2-simplex), you draw a line between all 3 points, forming a triangle. While a point is a 0-simplex. The TDA features recorded are the $birth$ (formation of holes), $death$ (deformation or the closing of holes) in different dimensions as well as the persistence of features. Persistence is defined as the length of time it took a feature to die ($death - birth$). This means that if a point touches another point then one of the points/features has died. The $death$ is recorded with the radii value at which the points overlapped. In addition, due to all the shifts and changes, from the 1-Dimension and upwards, some features may appear - a new hole, and this feature is recorded as a $birth$. The $birth$ feature is the radii at which it appeared.

However, for our task, we only record holes, known as connected components which are the holes in the 0-Dimension. We only use connected components features because we are able to obtain more uniform features across our datasets. Additionally, to obtain these TDA features, we need at least a 2D matrix as input to get the point clouds needed to obtain the simplicial complexes. Lastly, due to the rigorous process of counting the $birth$ and $death$ of new features, TDA is both a data reduction and feature extraction tool, as the output trims the noise from the original data. The noise-resistant quality of TDA makes it able to extract the true structure of a dataset. Therefore, for our text classification task - *deepfake text detection*, TDA is able to extract linguistic structural features.

5.4 Topology-Aware Deepfake Text Detector

5.4.1 TopRoBERTa_pool

To build this TDA-infused RoBERTa model, we focus on the four layers needed to convert vanilla-RoBERTa to Topological-RoBERTa - (1) pre-trained weights of the RoBERTa model, (2) dropout layer with probability p=0.3, (3) Topological layer for calculating, and (4) Linear transformation layer. See Figure 5.2 for a flow chart describing the architecture of TopRoBERTa with the 4 layers.

To train our end-to-end Topological-RoBERTa model, we first fine-tune RoBERTa-

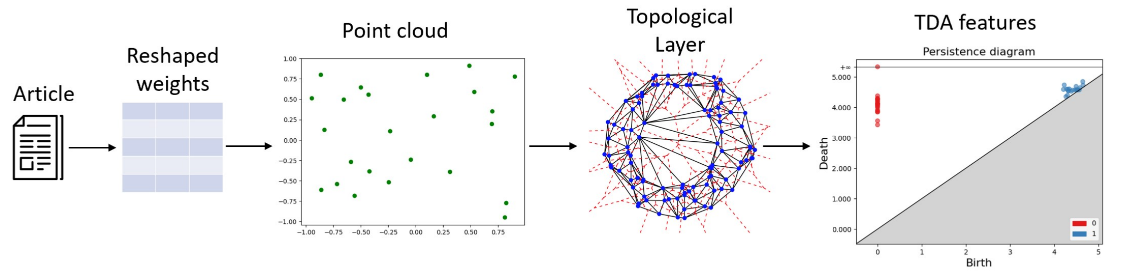


Figure 5.3: Illustration of how we extract the TDA features using the reshaped RoBERTa regularized weights as input. First, we reshape the regularized $pooled\_output$ from 1×768 dimensions to 24×32 and use this 2D matrix as input for the Topological layer. The Topological layer treats this 2D matrix as a point cloud plot and extracts TDA features ($birth$ & $death$). Next, these TDA features are plotted in a figure known as *Persistent Diagram*, where the $birth$ features are on the x-axis and $death$ features are on the y-axis. While we plot the features from the 0-Dimension (connected components) and 1-Dimension (loops), only 0-Dimension features are used for our classification task.

base model. As we fine-tune the model, we take the $pooled\_output$ which is a 1×768 vector containing the latent representations of RoBERTa. We find that RoBERTa weights are richer than BERT because it is a robustly trained BERT model and has over 20K more vocabulary size than BERT. These latent representations capture word-level and sentence-level relationships, thus extracting contextual representations [110]. Due to the contextual representations captured, RoBERTa weights essentially extract semantic and syntactic linguistic features [180, 181].

Next, we pass this $pooled\_output$ which is a 1×768 vector into a regularization layer, called $dropout$. This $dropout$ layer drops a pre-defined percentage (30% in our case) of our $pooled\_output$ to make our model more generalizable and less likely to overfit. The intuition behind this technique is that by dropping 30% of the $pooled\_output$, the rest of the 70% is forced to carry more weight and thus be less noisy. The advisable dropout probability is between 0.1 - 0.5, however, we use a probability of 0.3 as it has been shown to achieve good results on deepfake text detection [1]. Lastly, the dropped weights in our $pooled\_output$ are converted to zero so that the dimensions of the output $dropout(pooled\_output)$ is the same as the input - 1×768 vector.

Before, we use our regularized output - $dropout(pooled\_output)$ as input for the Topological layer[2], we first reshape it from 1D $\rightarrow$ 2D. TDA requires at least a 2D matrix to construct simplicial complexes that persistent homology technique uses to extract the

[2]https://github.com/aidos-lab/pytorch-topological/tree/main

$birth$ and $death$ of TDA features (connected components, specifically) [177]. This is because the simplicial complexes can only be extracted from the point cloud (which is a scatterplot of the dataset) and to get this point cloud we need a dataset with 2-coordinates. We include this Topological layer in RoBERTa because: (1) RoBERTa has richer latent representations than BERT [110]; (2) TDA is robust to noise, out-of-distribution, and limited data [178]; (3) TDA is able to capture more features that other feature extraction techniques cannot capture [175, 176]; and (4) TDA extracts the true structure of data points [177]. To convert the regularized weights from 1D $\rightarrow$ 2D is non-trivial because we need to get the best shape to obtain useful TDA features which are stable and uniform (vectors of the same length) across all input of a particular dataset. Therefore, we try different 2D sizes and find that the closer it is to a square matrix, the more stable the TDA features are. Stable in this context means that for every input, the TDA layer outputs the same number of features in a vector. Therefore, we convert the 1×768 vector to a 24×32 matrix, since it is the closest to a square matrix as 768 is not a perfect square. We also find through experimentation that when row $>$ column, TDA features are unstable. Unstable for our task means that the Topological layer output different vector sizes of TDA features given the input. Also, sometimes the feature vector can only contain nan values based on the input which means that it was unable to extract TDA features. Thus, we find that $pooled\_output$ must be reshaped such that row $\leq$ column and 24×32 satisfies this claim. Finally, using the 2D matrix as input to our Topological layer, we obtain the 0-Dimension features following the process illustrated in Section 5.3. This yields a 23×3. These 3 columns represent the $birth$ time, $death$ time, and persistence features, respectively. Persistence is defined as the length of time it took a feature to die. Next, this 2D matrix is flattened to a vector size of 1×69 so it can be easily concatenated with the 1D $dropout(pooled\_output)$. See Figure 5.3 for an illustration of how the TDA features are extracted. We interpret these TDA features as capturing linguistic structure, as it is capturing the structure and shape of textual data.

Lastly, we concatenate the regularized RoBERTa weights ($dropout(pooled\_output)$) of size 1×768 with the TDA features of size 1×69. This yields a vector of size 1×837. Thus, this 1×837 vector serves as input for the final layer of feature transformation, the Linear layer. The Linear layer's latent space increases from 768 to 837 in order to take the concatenated vector as input. TDA increases the latent space by 69 dimensions. However, we observe that unlike other TDA-based Transformer classifiers [24, 179], which use attention weights in which the size is dependent on the length of text, our TDA technique increases the latent space minimally. Finally, the output of this Linear layer is a vector that is the size of $batch\_size \times number\_of\_labels$. Thus, if we have $batch\_size = 16$ and $number\_of\_labels = 20$, we obtain a vector of size: 16×20. Finally, we pass this vector as input into the softmax layer for multi-class classification.

We train this Topological model for 5 epochs so both the RoBERTa pre-trained weights and Topological features can be improved. We use Adam optimizer, cross-entropy loss, and a learning rate of $2e - 5$. We experimented with different loss functions - cross-entropy, topological loss, contrastive loss, Gaussian loss, and different combinations of these loss functions. We find that just cross-entropy loss achieves the best performance. Finally, TDA features are compatible with non-TDA features [189], making it a suitable technique to extract subtle linguistic patterns that distinguish deepfake texts from human-written ones. Thus, **TopRoBERTa_pool** captures semantic, syntactic, and structural linguistic features.

5.4.2 TopRoBERTa_attn

TopRoBERTa_attn uses the same architecture as **TopRoBERTa_pool**, except we use the attention weights which is of size $Max\_length \times 768$ as input for the Topological layer. This technique is inspired by [24, 179] who use the directed and undirected graphs of the attention weights. Thus, instead of increasing the computational cost by building graphs with the attention weights, we use the attention weights as input for extracting the TDA features. This technique provides a fairer comparison to TopRoBERTa_pool.

Thus, using this large matrix as input, we obtain TDA features of size: $Max\_length-1 \times 3$ matrix. This 2D matrix is then flattened to a 1D vector of size $1 \times 3 * Max\_length$. However, we find that unlike TopRoBERTa_pool, TopRoBERTa_attn can be unstable. Unstable here means that the TDA feature vectors are not always the same length, so we had to implement normalization techniques to ensure that the TDA feature vectors are the same length. Additionally, we find that the feature vector changes per data, when we use the attention weights as input for the TDA layer because the maximum length is not fixed. For instance, if a dataset has a maximum length of 512 tokens, then the TDA feature space will be of size 511×3 which will be further flattened to 1×1533. Next we concatenate the $pooled\_output$ which is of size 1×768 with the TDA features, and obtain a vector of size 1×2301. This increases the latent space of the Linear layer from 768 to 2301. Finally, following the same procedure of TopRoBERTa_pool, we pass the Linear layer output into the softmax layer for multi-class classification.

5.5 Experiments

Dataset	Train	Validation	Test	# Labels
TuringBench	16K	5.4K	2.7K	20
SynSciPass	87K	10K	10K	12

Table 5.1: Dataset summary statistics

5.5.1 Datasets

Since the focus of this paper is multi-class authorship attribution of deepfake texts vs. human-written texts, we evaluated our models on two deepfake text datasets - TuringBench [1] and SynSciPass [63]. Furthermore, these datasets are a reflection of the real world, where we currently have more human-written texts examples than deepfake texts.

TuringBench dataset is a news (mostly politics) dataset comprised of both human-written and deepfake-generated texts. It has 20 labels - 1 human & 19 deepfake text generators. These 19 deepfake labels are generated using different pre-trained models of 10 unique generation model architectures. We cleaned the dataset further than the version used in the TuringBench paper [1] and skewed the dataset to maintain a more realistic real-world scenario. This skewed version contains 100% of the human examples and only 10% of each of the 19 deepfake examples. See Table 5.1 for the train, validation, and test splits of TuringBench dataset.

SynSciPass dataset is comprised of scientific articles, authored, by both human and deepfake authors. In addition to being grossly imbalanced, the SynSciPass dataset is noisy. This is because, unlike the TuringBench dataset where all the deepfake texts are generated with open-ended text-generators like GPT-3, SynSciPass's deepfake labels are generated with 3 types of text-generators. These are open-ended generators, translators like Google translate (e.g. English $\rightarrow$ Spanish $\rightarrow$ English), and paraphrasers like SCIgen and Pegasus. Using these different text-generation techniques introduces a noisiness in this dataset. We use the 12 labels - 1 human & 11 deepfake text-generators. However, due to the different NLG methods employed, this data also has 4 heterogeneous labels - human, generators, translators, and paraphrasers. See Table 5.1 for the train, validation,

MODEL	Precision	Recall	Accuracy	Macro F1	Weighted F1	% Gain
BERT	0.6884	0.7066	0.8058	0.6842	0.7891	-
Gaussian-BERT	0.6820	0.6871	0.7871	0.6832	0.7862	0.1% $\downarrow$
TopBERT_attn	0.7297	0.7265	0.8177	0.7137	0.8081	3% $\uparrow$
TopBERT_pool	0.7264	0.7265	0.8177	0.7178	0.8115	4% $\uparrow$
RoBERTa	0.7185	0.7162	0.8098	0.7030	0.8010	-
Gaussian-RoBERTa	0.6582	0.6629	0.7706	0.6572	0.7673	4% $\downarrow$
TopRoBERTa_attn	0.5276	0.3625	0.3405	0.2650	0.3186	39% $\downarrow$
TopRoBERTa_pool	**0.7366**	**0.7386**	**0.8249**	**0.7205**	**0.8130**	2% $\uparrow$

Table 5.2: TuringBench Authorship Attribution results. The best performance is **boldened** and the second best is underlined. The percentage gains reported in the *% Gain* are calculated from the Macro F1.

MODEL	Precision	Recall	Accuracy	Macro F1	Weighted F1	% Gain
BERT	0.8585	0.8148	0.9791	0.8327	0.9785	-
Gaussian-BERT	0.8404	0.7709	0.9745	0.7933	0.9735	4% $\downarrow$
TopBERT_attn	0.8022	0.8022	0.8080	0.8001	0.9799	3% $\downarrow$
TopBERT_pool	0.8682	0.8298	0.9807	0.8471	0.9802	2% $\uparrow$
RoBERTa	0.9012	0.8554	0.9853	0.8719	0.9846	-
Gaussian-RoBERTa	0.8929	0.8809	0.9872	0.8847	0.9870	1% $\uparrow$
TopRoBERTa_attn	0.9154	0.8826	0.9879	0.8923	0.9875	2% $\uparrow$
TopRoBERTa_pool	**0.9177**	**0.8978**	**0.9892**	**0.9058**	**0.9890**	4% $\uparrow$

Table 5.3: SynSciPass Authorship Attribution results. The best performance is **boldened** and the second best is underlined. The percentage gains reported in the *% Gain* are calculated from the Macro F1.

and test splits of SynSciPass dataset. The human label has 79K examples, while the rest of the labels have between 600-850 examples. Finally, since this dataset looks at the task from a different perspective, as well as being grossly imbalanced, we evaluated our models on these constraints.

5.5.2 Authorship Attribution

We train all the models with the same hyperparameters & parameters - dropout probability p=0.3, learning rate of $2e-5$, cross-entropy loss, batch size of 16 and 5 epochs. Also, tested with other loss functions (contrastive loss, topological loss, and Gaussian loss) and found cross-entropy to be the best. See models:

- **BERT:** We use BERT-base cased pre-trained model.
- **TopBERT_attn:** We add a Topological layer to the BERT model described above and follow the process described in Section 5.4.2.
- **TopBERT_pool:** We add a Topological layer to the BERT model described above and follow the process described in Section 5.4.1 and Figure 5.2.
- **Gaussian-BERT:** This is a BERT-base model with a Gaussian layer to add Gaussian noise to the weights. The hypothesis is that if TopBERT achieves superior performance randomly then adding a Gaussian layer should have a similar effect.
- **RoBERTa:** We use RoBERTa-base pre-trained model.
- **TopRoBERTa_attn:** We add a Topological layer to the RoBERTa model described above and follow the process described in Section 5.4.2.
- **TopRoBERTa_pool:** We add a Topological layer to the RoBERTa model described above and follow the process described in Section 5.4.1 and Figure 5.2.
- **Gaussian-RoBERTa:** This is similar to the *Gaussian-BERT* architecture, except we use RoBERTa-base instead of BERT-base. The hypothesis also remains the same here but applied to TopRoBERTa.

We only compare BERT and RoBERTa model variants to our deepfake text detection models to show the robustness of the models, as they are still the SOTA models. Furthermore, we did not compare other AA models as they have been shown to fail in the task of deepfake text detection [1, 190]. In fact, when such AA models outperform RoBERTa, they only achieve only a 1% increase in performance for our task.

Lastly, we evaluated the performance of these AA models with established evaluation metrics for machine learning - Precision, Recall, Accuracy, Macro F1 score, Weighted F1 score. However, due to the imbalanced nature of the datasets, we focus on the macro F1 score and use it to calculate positive and negative percentage gains for the classification task.

MODEL	Precision	Recall	Accuracy	Macro F1	Weighted F1	% Gains
BERT	0.9576	<u>0.9264</u>	<u>0.9895</u>	<u>0.9414</u>	<u>0.9894</u>	-
BERT_pool	<u>0.9616</u>	0.9221	<u>0.9895</u>	0.9412	0.9893	0.02% $\downarrow$
RoBERTa	0.9601	0.8767	0.9869	0.9064	0.9857	-
TopRoBERTa_pool	**0.9865**	**0.9638**	**0.9960**	**0.9746**	**0.9959**	7% $\uparrow$

Table 5.4: SynSciPass Authorship Attribution results with 4 labels - Human vs. Generators vs. Translators vs. Paraphrasers. The best performance is **boldened** and the second best is <u>underlined</u>. The percentage gains reported in the *% Gain* are calculated from the Macro F1.

5.6 Results

Our proposed models - TopBERT_pool and TopRoBERTa_pool are evaluated on their ability to more accurately attribute human- vs. deepfake-authored articles to their true authors. We specifically used an imbalance dataset - TuringBench and a noisy & imbalanced dataset - SynSciPass. From Tables 5.2 and 5.3, we observe that TopRoBERTa_pool excels in the AA task, consistently outperforming all other models. TopRoBERTa_pool outperforms RoBERTa by 2% and 4% for TuringBench and SynSciPass, respectively. Similarly, TopBERT_pool outperform BERT by 4% and 2% for TuringBench and SynSciPass, respectively. While TopRoBERTa_attn underperforms RoBERTa by 39% for TuringBench and outperforms RoBERTa by 2%. Also, TopBERT_attn is observed to outperform BERT by 3% for the TuringBench dataset, and underperform by 3% for the SynSciPass dataset. Furthermore, we observe that Gaussian-BERT underperforms BERT for both datasets. However, Gaussian-RoBERTa underperforms RoBERTa for TuringBench and outperforms RoBERTa for SynSciPass. This suggests that only TopBERT_pool and TopRoBERTa_pool consistently outperform their base models, given the constraints - noise and imbalance.

5.7 Further Analysis of TopRoBERTa_pool

The results in Tables 5.2 and 5.3 suggest that the addition of a Topological layer improves performance for our main contribution, the TDA $pooled\_output$ technique. More specifically in Table 5.3, this improvement is exaggerated, with TopRoBERTa_pool, achieving a 4% increase in macro F1 score for the SynSciPass dataset. To understand these improve-

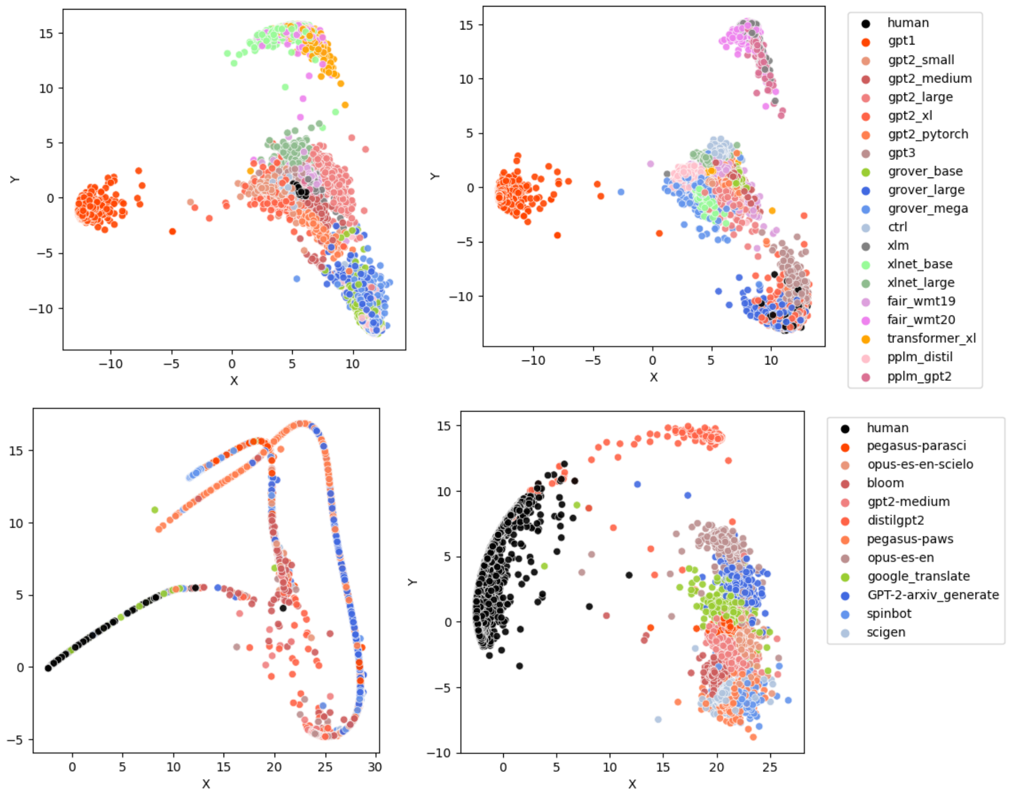


Figure 5.4: PCA plots from RoBERTa vs. TopRoBERTa training embeddings - TuringBench (above) & SynSciPass (below). For all plots, the **black** clusters are the human label and the other clusters are the deepfake labels.

ments further, we compare the weights of vanilla RoBERTa and TopRoBERTa_pool. This is done by plotting the PCA features of vanilla RoBERTa vs. TopRoBERTa_pool weights for the TuringBench and SynSciPass dataset. The hypothesis here is that if TDA captures more features than the base Transformer-based models, there will be a distinct or at least a subtle difference in the structure of these two models. In addition, due to the superior performance of TopRoBERTa_pool, we should also, observe more distinct clusters of the multiple authors with the Topological model than with the vanilla or base model. See Figure 5.4 for the PCA plots of the weights of RoBERTa and TopRoBERTa. For TuringBench, we observe that the shapes are similar with minimal differences in structure, however, there are more distinct clusters for TopRoBERTa. Next, for the SynSciPass dataset, TopRoBERTa shows not only more distinct clusters but also a very different shape from RoBERTa's PCA plot. These more distinct clusters observed from TopRoBERTa for both datasets could further explain the superior performances of

TopRoBERTa which is further discussed in Section 5.8.

Lastly, to further show the robustness of TopRoBERTa_pool to noisy data with heterogeneous labels, we run further experiments to compare vanilla Transformer-based models with Topological Transformer-based models. For this task, there are 4 labels of the SynSciPass dataset - Humans vs. Generators vs. Translators vs. Paraphrasers. Heterogeneous labels refer to a situation where the labels in a classification task are diverse and encompass multiple distinct categories or types. Due to the noisy and heterogeneous nature of the dataset with the 4 labels, we observe up to 7% increase in performance from TopRoBERTa_pool. However, TopBERT_pool performs similarly as BERT for this task, further confirming the superiority of TopRoBERTa_pool.

5.8 Discussion

See below for the observed strengths and weaknesses of adding a Topological layer to a Transformer-based model:

1. **Deepfake text has a shape which TDA captures:** The ultimate goal of TDA is to capture the true shape of data which is what TopRoBERTa_pool does for the TuringBench and SynSciPass dataset. These subtle and distinct structural differences observed between RoBERTa and TopRoBERTa_pool's PCA plots in Figure 5.4 confirm this. This phenomenon is supported by [175, 176, 189]'s findings that TDA captures features that other feature extractors are unable to capture. However, since text takes on the shape of its numerical representer, the structural features that TDA extracts from textual data are dependent on the numerical representer. Therefore, we can only claim that TDA extracts the shape of text data as best it can, given the input. This means that TDA captures the shape of deepfake texts as accurately as it can, given the BERT and RoBERTa weights. We also understand that using PCA for dimension reduction inherently loosely some information as well. Therefore, while we claim that TDA captures the structure of data, we do not claim that the PCA plots are the true or accurate shapes of each author's writing style. Furthermore, while structural features are important for accurate authorship attribution, text does not have an intuitive shape, so there is currently no technique to intuitively visualize the TDA features from text data that represent the shape of an author's writing style.

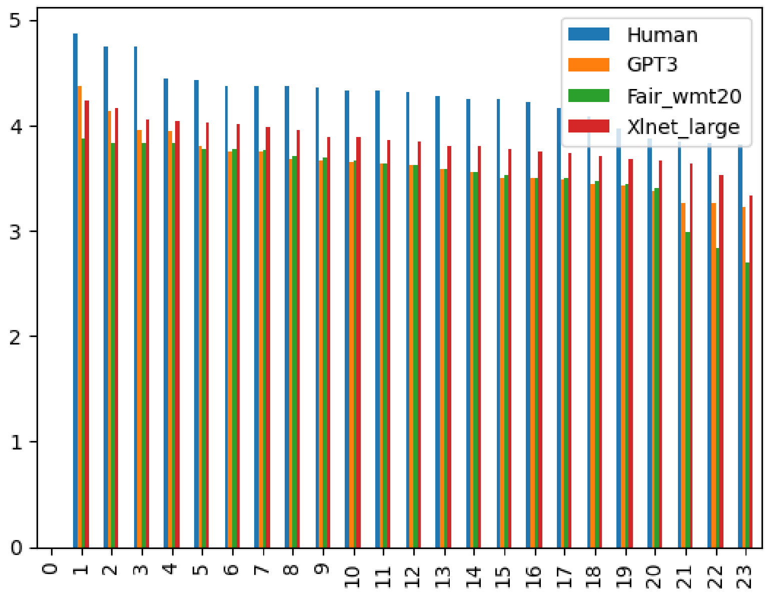


Figure 5.5: Plot of the $death$ features of authors - Human, GPT-3, FAIR_wmt20, and XLNet_large. All authors are considered very good writers, except for XLNet_large. We show that TDA is capturing distinct features that complement BERT and RoBERTa weights which improve the classification. There are 23 $death$ features from the TDA $pooled\_output$ technique as the input for extracting TDA features are 24×32 which gives us 24 point clouds, yielding 23 non-trivial TDA features for $birth$, $death$, and persistent columns.

2. **TDA is robust to noise, imbalanced, and heterogeneous datasets:** One of the benefits of TDA is that the persistent homology technique is a process to discover persistent features, such that noise is ignored, making TDA noise-resistant [177]. This is evident by the 4% increase in macro F1 (from RoBERTa-87% to TopRoBERTa_pool-91%) for the SynSciPass dataset which is grossly imbalanced, noisy, and heterogeneous. Heterogenous means that the SynSciPass has diverse categories, which is why it can be further distinguished by 4 labels - Humans vs. Generators vs. Translators vs. Paraphrasers. We observe a 7% increase in performance from TopRoBERTa_pool when classifying with these 4 labels (See Table 5.4). This further confirms the robustness of TDA to noise and heterogeneous labels [178]. Intuitively, we observe only a 2% increase in TopRoBERTa_pool's performance on the TuringBench dataset because although it is also grossly imbalanced, it is not noisy.

3. **TDA's structural features complement RoBERTa weights:** There are currently 5 linguistic levels - phonology, pragmatics, morphology, syntax, and semantics. And due to the complex nature of language, they may not be enough to capture an author's unique writing style. However, RoBERTa captures a broad range of linguistic patterns, such as contextual representations in terms of syntactic and semantic relationships. However, using only syntactic and semantic linguistic features is not sufficient to achieve accurate authorship attribution. Therefore, with TDA, additional features are extracted to push us closer to accurate authorship attribution. TDA features capture the shape and structure of data. These features in the context of NLP, can be interpreted as linguistic structures. Finally, by combining these 3 linguistic features - syntactic, semantic, and structural linguistic features, our model - **TopRoBERTa_pool** is able to more accurately distinguish deepfake texts from human-written ones, as observed in Tables 5.2, 5.3 and 5.4. Furthermore, in Figure 5.5, we observe that TDA captures some distinctness in terms of the persistence of the features between authors, which suggests that TDA captures additional features that complement RoBERTa.

4. **The quality of TDA features depend on input:** The performance difference between TopBERT and TopRoBERTa indicates the significance of the initial input for extracting valuable TDA features. To build an effective Topological-based text classification model, a reliable numerical representation of the texts is crucial. Since Transformer-based models like BERT and RoBERTa are still considered state-of-the-art, they remain the optimal choices. Consequently, our Topological-RoBERTa (TopRoBERTa_pool) outperforms Topological-BERT (TopBERT_pool) because RoBERTa extracts richer features compared to BERT for this task. This can be attributed to RoBERTa being a robustly trained BERT model with 20K additional tokens, enabling it to capture stronger linguistic features. This is potentially why TopRoBERTa_pool outperforms RoBERTa by a 7% margin on detecting heterogeneous labels and TopBERT_pool performs about the same as BERT in Table 5.4. Additionally, we find that to obtain stable and useful features, the better input for our Topological layer is a 2D matrix of size row $\leq$ column of the $pooled\_output$. Stable features refer to obtaining the same number of features for each article, ensuring a consistent vector size. Lastly, when using TDA features extracted with attention weights, we observe that TopRoBERTa_attn performs comparably to TopRoBERTa_pool for SynSciPass but underperforms for TuringBench.

This inconsistency further supports our findings that TDA is sensitive to input, emphasizing the need for better numerical representations of texts.

5. **Improvement with TDA is not random:** As RoBERTa is a black-box model, to confirm that TopRoBERTa's performance is not due to training with the right kind of "noise," we compare with Gaussian models - Gaussian-BERT & Gaussian-RoBERTa. The hypothesis is that if TopRoBERTa's performance is due to noise, then Gaussian models trained on noise should perform similarly. We observe from Tables 5.2 & 5.3 that Gaussian-BERT underperforms BERT for both datasets, however Gaussian-RoBERTa underperforms RoBERTa for TuringBench but outperforms on the SynSciPass dataset. Furthermore, TopBERT_attn and TopRoBERTa_attn are also observed to perform inconsistently like the Gaussian models. While TopBERT_pool and TopRoBERTa_pool consistently outperform their base models. Thus, due to the consistency in performance from TopRoBERTa_pool and TopBERT_pool, it suggests that their improvement is not random.

6. **Reshaped pooled output is a better TDA input than attention weights:** Most researchers that apply TDA for NLP tasks commonly use word2vec embeddings or attention maps as input to extract TDA features [24, 179, 183–186]. However, we use reshaped pooled outputs because while using the attention weights as input is more intuitive, the $pooled\_output$ contains richer features for classification than attention weights. Furthermore, we also discover in confirmation with [24] that using attention weights to extract TDA features is unstable. This could potentially be the reason why previous studies first construct a directed and undirected graph with the attention weights prior to TDA feature extraction to encourage stability [24, 179]. Nevertheless, these techniques significantly increase computational costs. The attention weights are large matrices, such as 400×768 for TuringBench and 512×768 for SynSciPass datasets. After TDA feature extraction, the flattened vectors are 1×1197 and 1×1533 for the TuringBench and SynSciPass datasets, respectively. These increase the latent space of the Linear layer by a large margin - 768 to 1965 and 2301 for the TuringBench and SynSciPass datasets, respectively. Additionally, to maintain consistent dimensions for TDA features, we employ normalization techniques when TDA features for some articles differ from the majority. In contrast, our main technique increases the dimensions of the linear layer to 837 for both datasets without requiring normalization. The inconsistent performance

of TopBERT_attn and TopRoBERTa_attn further supports the superiority of using $pooled\_output$ as input for obtaining TDA features compared to attention weights.

7. **TopBERT_attn and TopRoBERTa_attn performance are inconsistent because attention weights are length-dependent:** Unlike the $pooled\_output$, the attention weights are length-dependent because attention weights are word-level representations while $pooled\_output$ are sentence-level representations. This is because each row of the attention weights matrix is a 1×768 vector representing each word in a text, such that if the max length is 200, the attention weight is of size - 200×768. Thus, when this attention weight is used as input for the Topological layer, the TDA features take in length, extracting structural length-based features. However, as articles, both news and scientific tend to have varying lengths, length features become a weak feature. Furthermore, as mentioned above, the Topological layer is sensitive to input, and since attention weights are not a good feature for *deepfake text detection*, TopBERT_attn and TopRoBERTa_attn perform inconsistently.

5.9 Summary

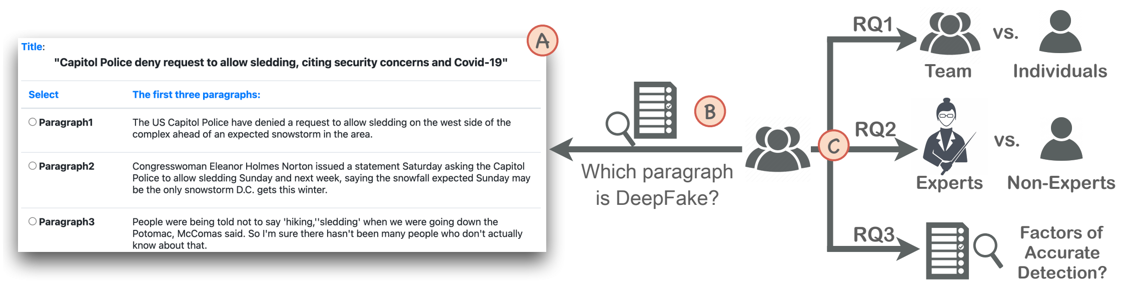
In conclusion, we propose a novel solution to accurately attribute the authorship of deepfake texts vs. humans. This novel technique entails including a Topological layer to RoBERTa-base model, such that the Linear layer's input is a concatenation of the RoBERTa regularized weights and the TDA features. We showed 2 techniques to extract TDA features with RoBERTa - TopRoBERTa_pool and TopRoBERTa_attn. We find that our novel technique - TopRoBERTa_pool is superior. Furthermore, we evaluated our models on noisy, grossly imbalanced, and heterogeneous datasets. We observe that TopRoBERTa_pool consistently outperforms all other models. Also, to investigate if our TopRoBERTa_pool's superior performance is random, we compare it to Gaussian models and find that in fact the Gaussian models are inconsistent, while our model remains consistent. Lastly, in the future, we would scrutinize our models under stricter constraints such as evaluation on adversarial robustness, known as *Authorship Obfuscation*, and out-of-distribution datasets, such as low-resource languages, multi-lingual, multi-domain, imbalanced, and insufficiently sized datasets.

Chapter 6 | Understanding Individual and Team-based Human Factors in Detecting Deepfake Texts

6.1 Introduction

In this work, we investigate the task of determining if a given text is a deepfake text or not. While the field of open-ended text generation is still relatively new, both computational and non-computational detection of deepfake texts have been extensively studied in recent years and well surveyed in [174]. In particular, what we are interested in answering is *how "humans" are able to detect deepfake texts better*. Clearly understanding human capacity and their limitations in detecting deepfake texts would help the development of both computational and non-computational (and even hybrid) tools for detecting deepfake text better. Recent literature (e.g., [1, 31, 191, 192]) has shown that, by and large, humans are *not* good at detecting deepfake texts, performing only slightly better than the level of random guessing. Even if humans are trained to detect deepfake texts, the performance has not improved significantly (e.g., [23, 118, 191]).

Our task is modeled similarly to the *Turing Test* problem defined by Alan Turing in the 1950s. The *Turing Test* is a test administered by a human as the judge who has a conversation with an unknown entity and decides if they are speaking with a human or machine/AI model. If the machine is labeled as human, then the machine has passed the test. However, for our task, due to the security risks deepfake texts pose, it is imperative that the NTG does not pass the test. To that end, we propose a framework that increases



Title:
"Capitol Police deny request to allow sledding, citing security concerns and Covid-19"

Select	The first three paragraphs:
○ Paragraph1	The US Capitol Police have denied a request to allow sledding on the west side of the complex ahead of an expected snowstorm in the area.
○ Paragraph2	Congresswoman Eleanor Holmes Norton issued a statement Saturday asking the Capitol Police to allow sledding Sunday and next week, saying the snowfall expected Sunday may be the only snowstorm D.C. gets this winter.
○ Paragraph3	People were being told not to say 'hiking,''sledding' when we were going down the Potomac, McComas said. So I'm sure there hasn't been many people who don't actually know about that.

Figure 6.1: (A) Example of a multi-authored (Human & Deepfake) 3-paragraph article; (B) Task: Detecting Deepfake texts; (C) Description of three research questions. "Image from [3]."

the probability of GPT-2 failing the *Turing Test*.

Therefore, in this work, we aim to find ways to improve humans in detecting deepfake texts better and understand human factors at play. Especially, we wonder if individual humans, trained or not, are not good at detecting deepfake texts, does their collaboration or expertise matter? As such, we pose the following three research questions (RQs):

RQ1 Do collaborative teams/groups perform better than individuals in deepfake text detection?

RQ2 Do experts perform better than non-experts in deepfake text detection?

RQ3 What are the factors that maximize the performance gain?

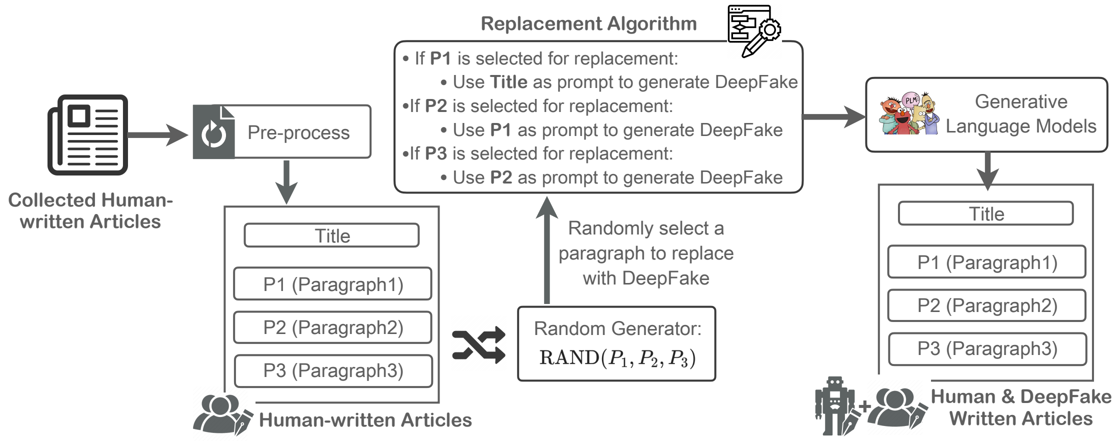
RQ1 aims to investigate what improves human performance from the baseline - team collaboration or individuals, and if collaboration improves human performance significantly, which collaboration technique matter - synchronous or asynchronous collaboration? The hypothesis here is that synchronous collaboration will improve human performance in deepfake text detection because humans perform better when there is informal discussion and sharing of ideas, as shown in prior literature (e.g., [193], [194], [195]). Given the benefits of collaboration, we hypothesize that collaboration will improve human performance. **RQ2** aims to investigate how English experts vs. English non-experts detect deepfake texts differently. English experts are defined as individuals with at least a Bachelor's degree in English (and related programs). We aim to investigate the characteristics that make one or more settings significantly outperform others. An example here is that we hypothesize that experts will focus more on high-level errors such as logical fallacies and non-experts will focus more on low-level errors such as grammar issues.

RQ3 aims to investigate the human factors that may help improve human performance in deepfake text detection. See Figure 6.1(C) for a visual representation of the research questions.

Finally, our main contribution is investigating human participants' ability to detect deepfake texts with different settings – non-expert vs. expert and individual vs. collaborative. Our key findings are summarized as follows: (1) both expert and non-expert in the individual settings outperform the baseline significantly; (2) experts improve significantly from individual to collaborative settings; and (3) experts more frequently use strong indicators of deepfake texts as justification (which explains their superior performance).

6.2 Methodology

To improve human performance in deepfake text detection, we first, define a realistic problem - detecting deepfake texts in an article authored by both human and an AI (i.e., GPT-2). This is done by randomly replacing 1 out of 3 human-written paragraphs with a GPT-2-generated paragraph. Next, we define 2 variables for our study - *individual vs. collaboration* and *non-expert vs. expert*. After using these variables to facilitate crowdsourcing recruitment, we ask the human participants to select 1/3 paragraphs that is deepfake and provide justification for selection.



Replacement Algorithm
• If **P1** is selected for replacement:
• Use **Title** as prompt to generate DeepFake
•If **P2** is selected for replacement:
• Use **P1** as prompt to generate DeepFake
•If **P3** is selected for replacement:
• Use **P2** as prompt to generate DeepFake
Collected Human-written Articles
Pre-process
Generative Language Models
Title
P1 (Paragraph1)
P2 (Paragraph2)
P3 (Paragraph3)
Human-written Articles
Randomly select a paragraph to replace with DeepFake
Random Generator: RAND(P_1, P_2, P_3)
Title
P1 (Paragraph1)
P2 (Paragraph2)
P3 (Paragraph3)
Human & DeepFake Written Articles

Figure 6.2: Illustration of the data generation process. "Image from [3]."

6.2.1 Data Generation

We first define the goal of distinguishing deepfake texts from human-written texts. As this problem has been studied by several researchers [2, 15, 117, 191, 196] and shown to be very non-trivial and difficult to solve, we develop a novel way of administering the *Turing Test*. Thus, we implement a realistic setting of the problem - *detecting deepfake texts in an article authored by both humans and an NTG*. This setting is motivated by the fact that while NTGs currently have very impressive generations, humans still produce more natural speech than NTGs. This means that it will be natural for humans to edit machine-generated articles to make them sound more authentic. We study this non-trivial problem by asking human evaluators to: (1) Of 3 paragraphs, two written by a human, and one machine-generated, select the machine-generated paragraph. (2) Please check all explanations that satisfy the reason(s) for your choice. See Figure 6.1(B) for a visual representation of our task. We provide seven pre-defined rationales that correspond to flaws typically observed in deepfake text [192] - grammatical issues, repetition, lacks common sense, contains logical errors, contradicts previous sentences, lack of creativity or boring to read, writing is erratic (i.e. does not have a good flow) or choose to write on their own.

To build this dataset, we collected 200 human-written news articles (mostly politics since this work is motivated by mitigating the risk of mis/disinformation or fake news dissemination) from reputable news sources such as CNN and Washington Post. Next, of the 200 articles, we took the first 50 articles with at least 3 paragraphs. Then, we removed all paragraphs after the 3rd paragraph. Since the goal is to have a multi-authored article (human and AI), we randomly select 1/3 paragraphs to be replaced by GPT-2's [38] generated texts. We used only GPT-2 to generate the deepfake texts because: (1) GPT-2 and GPT-3 are similar. Based on [15, 191], human performance on detecting GPT-2 and GPT-3 texts have similar accuracies; and (2) GPT-2 is cheaper to generate texts with than GPT-3 since GPT-2 is open-source and GPT-3 is not. For generation, we used GPT-2 XL which has 1.5 billion parameters and *aitextgen*[1], a robust implementation of GPT-2 to generate texts with the default parameters. We followed the following replacement process:

1. If paragraph 1 is to be replaced: Use Title as a prompt to generate GPT-2 replacement

[1]https://github.com/minimaxir/aitextgen

Label	Paragraph1	Paragraph2	Paragraph3
Count	23	16	11

Table 6.1: Data labels of deepfake texts.

2. If paragraph 2 is to be replaced: Use Paragraph 1 as a prompt to generate GPT-2 replacement

3. If paragraph 3 is to be replaced: Use Paragraph 2 as a prompt to generate GPT-2 replacement

Since we are unable to control the number of paragraphs GPT-2 generates given a prompt, we use a Masked Language model to choose the best GPT-2 replacement that fits well with the article. We use a BERT-base [82] as the Masked Language model to get the probability of the next sentence. Let us call this model G(.), it takes 2 inputs - the first and probable second sentence/paragraph (G($Text\_1, Text\_2$)) and outputs a score. The lower the score, the more probable $Text\_2$ is the next sentence.

For instance, say GPT-2 texts is to replace Paragraph 2 (P2) of an article

- We use P1 as prompt to generate P2 with GPT-2
- GPT-2 generates another 3-paragraph article with P1 as the prompt
- To find the suitable P2 replacement, we do G(P1, each GPT-2 generated paragraph)
- Since low scores with G(.) is considered most probable, the P2 replacement is the GPT-2 paragraph that yielded the lowest score with G(.)

We use a random number generator to select which paragraphs are to be replaced and got the following deepfake text replacement for the paragraphs in Table 6.1. See Table 6.2 for the average count of characters, words, and sentences in each of the 3 paragraphs. After we created these multi-authored articles, we manually did a quality check of a few of these articles by checking for consistency and coherence. See Figures 6.2 for the data generation process and 6.1(A) for an example of the final multi-authored article.

Next, as we have defined this realistic scenario, we hypothesize that collaboration will improve human detection of deepfake texts. Thus, we define 2 variables for this experiment - Individual vs. Collaboration and English expert vs. English non-expert. We investigate how collaboration (both synchronous and asynchronous) improves from

Label	Paragraph1	Paragraph2	Paragraph3
# Char	231.64	261.12	260.64
# Word	38.42	44.06	43.98
# Sentence	2.86	2.96	3.20

Table 6.2: Summary statistics of the articles

individual-based detection of deepfake texts. The hypothesis here is that when humans come together to solve a task, collaborative effort will be a significant improvement from average individual efforts. Additionally, as human detection of deepfake texts is non-trivial, we want to investigate if the task is non-trivial because English non-experts focus on misleading cues as opposed to English experts.

Finally, we observe that based on the replacement algorithm, some bias in detection may be introduced. Replacing paragraph 3 may be seen as easier because there is no other paragraph after it to judge the coherency. However, we keep the generation process fair by only using the text right before the paragraph as a prompt to generate the next paragraph. Thus, to replace paragraph 3, we only use paragraph 2 as a prompt, not the previous paragraphs and title.

6.2.2 Participant Recruitment

6.2.2.1 AMT

Inspired by [191], [117], and [197], we used Amazon Mechanical Turk (AMT) to collect responses from non-expert evaluators. We deployed a two-stage process to conduct the non-expert human studies. First, we posted a *Qualification* Human Intelligence Task (HIT) that pays $\$0.50$ per assignment on MTurk to recruit 240 qualified workers In terms of the qualification requirements, in addition to our custom qualification used for worker grouping, three built-in worker qualifications are used in all the HITS, including *i)* HIT Approval Rate ($\leq 98\%$), Number of Approved HITs (≥ 3000), and Locale (US Only) Qualification.

Next, we only enabled qualified workers to enter the large-scale labeling tasks. The approximate time to finish each labeling task is around 5 minutes (i.e the average time of two authors on finishing a random HIT). Therefore, we aim for $\$7.25$ per hour and set the final payment as $\$0.6$ for each assignment. Further, we provide "double-payment"

Participant	**Gender**	**Education**	**Group**
P1	Female	Bachelor's degree	G1
P2	Female	Bachelor's degree	
P3	Female	Bachelor's degree	
P4	Female	Bachelor's degree	G2
P5	Male	Bachelor's degree	
P6	Male	Graduate degree	
P7	Female	Graduate degree	G3
P8	Female	Graduate degree	
P9	Female	Bachelor's degree	
P10	Female	Bachelor's degree	G4
P11	Female	Bachelor's degree	
P12	Male	Bachelor's degree	
P13	Female	Graduate degree	G5
P14	Female	Bachelor's degree	
P15	Male	Graduate degree	
P16	Female	Bachelor's degree	G6
P17	Male	Bachelor's degree	
P18	Male	Bachelor's degree	

Table 6.3: Upwork participant demographics.

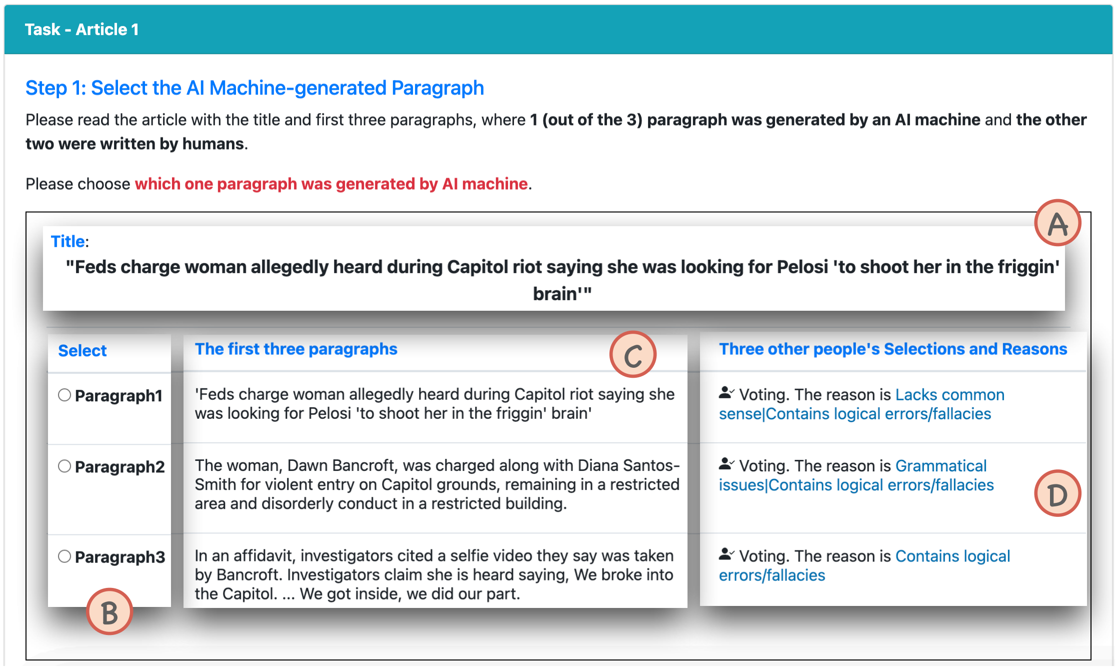
to workers who made correct submissions as an extra bonus to motivate non-expert participants to make better predictions.

6.2.2.2 Upwork

We employed Upwork as a platform to enlist proficient evaluators, particularly those with specialized knowledge in writing domains. Upwork is a prominent freelance website with an extensive network, generating an average of 40 million monthly visits[2]. It has revolutionized the freelance industry by offering access to skilled freelancers in various fields such as writing, graphic design, and web development. Using its automated recommendation system, Upwork can effectively match clients and workers based on their respective requirements.

On UpWork, we initially posted a job containing the description of our research problem and detailed information about what we are looking for in a participant - at least 18 years and an English native speaker. Next, we get more specific in our questions: (1)

[2]`https://sellcoursesonline.com/Upwork-statistics`


Task - Article 1

Step 1: Select the AI Machine-generated Paragraph

Please read the article with the title and first three paragraphs, where **1 (out of the 3) paragraph was generated by an AI machine** and **the other two were written by humans**.

Please choose **which one paragraph was generated by AI machine**.

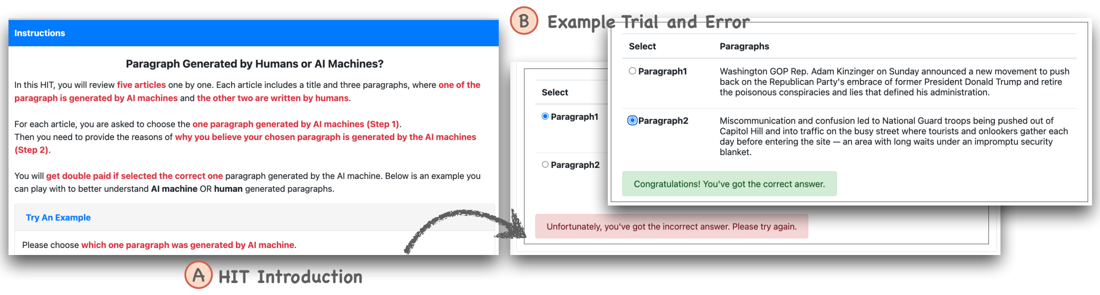
Title:
"Feds charge woman allegedly heard during Capitol riot saying she was looking for Pelosi 'to shoot her in the friggin' brain'"

Select	The first three paragraphs	Three other people's Selections and Reasons
○ Paragraph1	'Feds charge woman allegedly heard during Capitol riot saying she was looking for Pelosi 'to shoot her in the friggin' brain'	Voting. The reason is Lacks common sense|Contains logical errors/fallacies
○ Paragraph2	The woman, Dawn Bancroft, was charged along with Diana Santos-Smith for violent entry on Capitol grounds, remaining in a restricted area and disorderly conduct in a restricted building.	Voting. The reason is Grammatical issues|Contains logical errors/fallacies
○ Paragraph3	In an affidavit, investigators cited a selfie video they say was taken by Bancroft. Investigators claim she is heard saying, We broke into the Capitol. ... We got inside, we did our part.	Voting. The reason is Contains logical errors/fallacies

Figure 6.3: User Interface for the AMT Collaborative Group workers to choose the machine-generated paragraph. "Image from [3]."

What is the highest level of education you have attained? (2) Did you major in English or English Literature during your studies? (3) Can you share any recent experience you have had with projects similar to this?

To ensure that we recruited eligible participants, we assessed their age, language proficiency, and education by reviewing the information provided in their profiles. Additionally, we evaluated their proposal responses. Based on these criteria, we selected 18 finalists who were then invited to participate in the study. Prior to activating their Upwork contracts, we sent them the consent form through the platform's messaging function. We proceeded with the contract only after receiving the signed consent form from each participant. The primary purpose of these contracts was to enable clients to compensate workers for their submitted hours through the Upwork system. The hourly wages requested by the participants ranged from $25-35$, depending on their education levels and prior experience. We are pleased to report that all 18 participants successfully signed both documents and were compensated accordingly. For more details about the demographic breakdown of the recruited Upworkers, please refer to Table 6.3.



Instructions

Paragraph Generated by Humans or AI Machines?

In this HIT, you will review **five articles** one by one. Each article includes a title and three paragraphs, where **one of the paragraph is generated by AI machines** and **the other two are written by humans**.

For each article, you are asked to choose the **one paragraph generated by AI machines (Step 1)**.
Then you need to provide the reasons of **why you believe your chosen paragraph is generated by the AI machines (Step 2)**.

You will **get double paid if selected the correct one** paragraph generated by the AI machine. Below is an example you can play with to better understand **AI machine** OR **human** generated paragraphs.

Try An Example

Please choose **which one paragraph was generated by AI machine**.

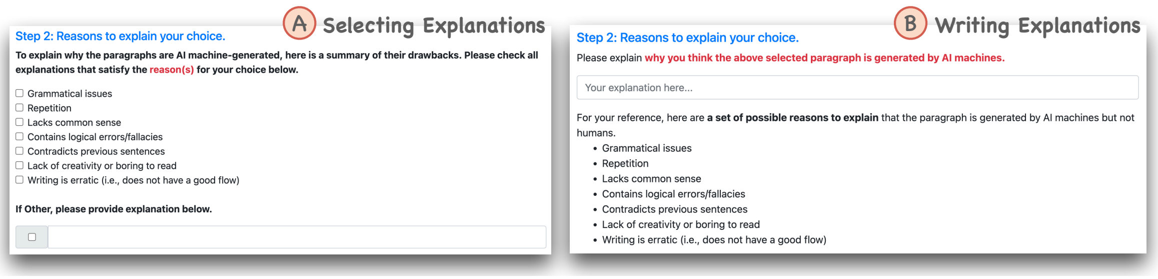
Select	Paragraphs
Paragraph1	Washington GOP Rep. Adam Kinzinger on Sunday announced a new movement to push back on the Republican Party's embrace of former President Donald Trump and retire the poisonous conspiracies and lies that defined his administration.
[✓ Paragraph2]	Miscommunication and confusion led to National Guard troops being pushed out of Capitol Hill and into traffic on the busy street where tourists and onlookers gather each day before entering the site — an area with long waits under an impromptu security blanket.

Congratulations! You've got the correct answer.

Select
[✓ Paragraph1]
Paragraph2

Unfortunately, you've got the incorrect answer. Please try again.

Figure 6.4: The instructions to train users by providing prompt feedback. "Image from [3]."



Step 2: Reasons to explain your choice.

To explain why the paragraphs are AI machine-generated, here is a summary of their drawbacks. Please check all explanations that satisfy the reason(s) for your choice below.

- ☐ Grammatical issues
- ☐ Repetition
- ☐ Lacks common sense
- ☐ Contains logical errors/fallacies
- ☐ Contradicts previous sentences
- ☐ Lack of creativity or boring to read
- ☐ Writing is erratic (i.e., does not have a good flow)

If Other, please provide explanation below.

☐

Step 2: Reasons to explain your choice.

Please explain **why you think the above selected paragraph is generated by AI machines.**

Your explanation here...

For your reference, here are **a set of possible reasons to explain** that the paragraph is generated by AI machines but not humans.

- Grammatical issues
- Repetition
- Lacks common sense
- Contains logical errors/fallacies
- Contradicts previous sentences
- Lack of creativity or boring to read
- Writing is erratic (i.e., does not have a good flow)

Figure 6.5: *Select* (A) vs. *Write* (B) justification question type. "Image from [3]."

6.2.3 Experiment Design

6.2.3.1 AMT

During the large-scale labeling task, we divide the recruited qualified workers into two groups to represent the individual vs. collaborative settings, respectively. We define group 1 as *Individual Group*, in which each worker was asked to select the machine-generated paragraph without any references. See Figure 6.3, for example, humans in *Individual Group* can only see the introduction with panels (A) (B), and (C). On the other hand, we design group 2 to be *Collaborative Group*, where the workers were asked to conduct the same task after the *Individual Group* finishes all HITs (i.e., see panel (A), (B), (C) in Figure 6.3). In addition, workers from the *Collaborative Group* could also see the selection results from the group 1 in an asynchronously manner, as the example shown in Figure 6.3(D), to support their own selection.

In addition, we investigate the capability of Individual vs. Collaboration of non-expect human participants to improve human performance in deepfake text detection. To do this, we compare 2 ways the human participants can provide justification for their answer - *Select* & *Write*. For the *select* setting, the question type was inspired by RoFT [117], a gamification technique for improving human performance in deepfake text detection. In the RoFT framework, participants were asked to select from a pre-defined list one or more reasons such as repetition, grammar errors, etc. Participants were also given another option, where they can enter their own justification if they do not find any suitable selection from the provided list. However, as this may be limiting because we pre-define the justifications, we also investigated another question type - *write*. In this setting, participants were asked to provide their reasoning. To help, we also share the list of justifications in the *select* setting to give the participants an idea of what justifications look like. See Figure 6.5 for *select* and *write* AMT interface.

Furthermore, we take actions to incentivize workers to provide qualified results: *i)* in our instruction, we provide immediate feedback on the worker's selection to calibrate their accuracy. In specific, after reading the HIT instruction (i.e., Figure 6.4 (A)), workers can get a deeper understanding of "which paragraph is generated by AI machine" by trial and error on selecting one example (i.e., Figure 6.4 (B)). Participants were given unlimited chances to change their answers. This example-based training process was inspired by [191]'s human evaluation study and was found to be the most effective training technique. *ii)* We pay double compensations to the workers who provide correct answers as mentioned in Section 6.2.2.1. This aims to encourage workers to get high accuracy on selecting the correct machine-generated paragraphs. *iii)* We set the minimum time constraint (*i.e.,* one minute) for workers to submit their HITs so that the workers will concentrate on the task for at least one minute instead of randomly selecting one answer and submitting the HIT. Note that we also disabled the copy and paste functions in the user interface to prevent workers from searching for the paragraphs from online resources.

6.2.3.2 Upwork

Our Upwork study comprises of two sub-experiments that aim to compare the accuracy of expert deepfake text detection in individual and collaborative settings. These two experiments, require participants to perform a task individually, and later on, perform the same task, collaboratively with 2 other participants in a synchronous way.

For the generation and dissemination of the study form, we used the Qualtrics[3] service. The form's user interface was equivalent to the *select* scenario presented in Figure 6.5.

[3] `https://www.qualtrics.com`

Upwork participants were given one week to complete the survey. Upon completion, we randomly assigned three participants per team, resulting in a total of six teams for synchronous collaboration (refer to Table 6.3). We conducted all discussions on the video communication software, Zoom[4], and utilized Zoom's built-in audio transcription feature, powered by Otter.ai[5], for discourse analyses. Prior to the start of each session, verbal consent was obtained from all participants for their participation in the discussion and audio recording, in addition to the written consent obtained during the recruitment procedure. One member of the study team acted as a moderator for the meetings. Depending on the participants' schedules and levels of commitment within their group, each meeting lasted between 1.5 to 3 hours.

6.2.4 Analysis Methods

To address RQ1 and RQ2, we begin by conducting a quantitative comparison of human participants' performance (Section 6.3.2). This involves measuring the mean accuracy of participants at the article level. Subsequently, we categorize the rationales provided by participants and compare their distributions based on correct and incorrect responses, which supports RQ3 (Section 6.3.3 & 6.3.4). Consistent with the analyses of RQ1 and RQ2, for RQ3, we compare the frequency of justifications across different settings such as *individual vs. collaboration* and *non-experts (AMT) vs. experts (Upwork)*. Finally, we outline the implications of our findings in Section 6.4.

6.3 Results

6.3.1 Performance Measurement

We evaluated the performance of participants in different experimental settings by measuring their accuracy in completing a set of 50 questions Q=q_1, q_2,..., q_{50}. The proportion of participants who answered a question correctly was calculated as acc_n =l_n/m_n * 100, where l_n is the number of participants who answered q_n correctly and m_n is the total number of participants who attempted q_n. This resulted in a list of accuracy scores ACC=acc_1, acc_2, ..., acc_{50}, representing the performance of participants across 50 articles. We used an unpaired independent sample T-test to compare the means of accuracy

[4]https://zoom.us
[5]https://otter.ai

SETTING	Select		Write	
	Mean Accuracy	p-value	Mean Accuracy	p-value
Baseline vs. Individual	32% vs. 44.99%	0.0004	32% vs. 45.92%	0.002
Baseline vs. Collaboration	32% vs. 51.35%	0.00007	32% vs. 51.97%	0.00003
Individual vs. Collaboration	44.99% vs. 51.35%	0.187	45.92% vs. 51.97%	0.26

Table 6.4: T-test Results for AMT Experiments (RQ1).

SETTING	Select	
	Mean Accuracy	p-value
Baseline vs. Individual	31% vs. 56.11%	9.1e-12
Baseline vs. Collaboration	31% vs. 68.87%	7.3e-15
Individual vs. Collaboration	56.11% vs. 68.87%	0.008

Table 6.5: T-test Results for Upwork Experiments (RQ1).

scores between different groups (individual vs. collaborative & non-experts vs. experts settings). To ensure the validity of the T-test, we conducted the Kolmogorov-Smirnov test and confirmed the normality assumption. Here, we present the results of the statistical testing.

6.3.2 Deepfake Text Detection Performance (RQ1 & RQ2)

6.3.2.1 RQ1: Individual vs. Collaboration

The AMT participants' baseline accuracy, which is equivalent to random guessing, is 32%. However, in the select and write settings, AMT-Individual improved their accuracy to 45% and 46%, respectively. On the other hand, AMT-Collaboration achieved 51% and 52% accuracy in the select and write settings, respectively. Both individual and team-based problem-solving in AMT experiments significantly outperformed random guessing, with a p-value less than 0.05 (Table 6.4). However, there was no significant difference in mean accuracy between individual and collaborative settings. In contrast, Upwork participants had a baseline accuracy of 31% and achieved an accuracy of 56% and 69% for Individual and Collaboration, respectively. All Upwork results were significant. For the average accuracy of AMT and Upwork participants, refer to Tables 6.4 and 6.5, respectively.

SETTING	Select		Write	
	Mean Accuracy	p-value	MeanAccuracy	p-value
AMT-Individual vs. Upwork-Individual	44.99% vs. 56.11%	0.005	45.92% vs. 56.11%	0.028
AMT-Collaboration vs. Upwork-Collaboration	51.35% vs. 68.87%	0.002	51.97% vs. 68.87%	0.003

Table 6.6: T-test Results for AMT vs. Upwork (RQ2).

6.3.2.2 RQ2: Non-Experts (*AMT*) vs. Experts (*Upwork*)

The study included both non-expert participants from AMT and expert participants from Upwork. In the Individual setting, non-experts achieved an accuracy of 45%, while experts achieved an accuracy of 56%. Similarly, in the Collaboration setting, non-experts achieved an accuracy of 51% compared to the experts' accuracy of 69%. The statistical analysis revealed that the differences between non-experts and experts in all settings were significant, with p-values < 0.05 for baseline vs. individual, baseline vs. collaboration, and individual vs. collaboration comparisons. Refer to Table 6.6 for a detailed T-test result of human performance on deepfake text detection between non-experts and experts.

6.3.3 Justification Patterns - *select* (RQ3)

Based on previous research [191, 192], we identified 7 types of justifications that participants could use to identify a paragraph as deepfake: *grammar issues, repetition, lacks common sense, contains logical errors/fallacies, contradicts previous sentences, lack of creativity or boring to read, writing is erratic/incoherent*. If none of these categories were applicable, participants were asked to provide an additional justification under the category of *other*.

To compare the frequency of justifications used by participants in different experimental conditions (individual vs. collaboration and non-experts vs. experts), we first computed the frequency of each justification category cited by each of the 4 groups. We then calculated the overall frequency of justifications used and the frequency of justifications used for correct and incorrect responses. Finally, we conducted an independent sample T-test to evaluate the statistical significance of the differences in the use of these justification categories.

Justification Type	Correct				Incorrect			
	AMT		Upwork		AMT		Upwork	
	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value
Grammar	13.97 vs. 23.08	0.016	15.33 vs. 24.6	0.013	15.65 vs. 16.89	0.675	14.22 v.s 12.07	0.469
Repetition	6.73 vs. 6.69	0.986	4 vs. 6.4	0.266	8.53 vs. 5.62	0.113	1.67 vs. 2	0.725
Common Sense	9.25 vs. 15.48	0.036	13 vs. 28	9.67e-05	13.02 vs. 9.94	0.206	3.33 vs. 5.56	0.171
Logical Errors	11.64 vs. 10.24	0.589	7.78 vs. 14.4	0.006	18.54 vs. 7.7	4.66e-05	3.89 vs. 4	0.942
Self-Contradiction	9.35 vs. 5.57	0.092	7.67 vs. 14.8	0.011	18.01 vs. 6.7	1.53e-06	6.56 vs. 3.6	0.054
Lack of Creativity	12.87 vs. 13.49	0.843	8.33 vs. 7.6	0.734	16.9 vs. 14.13	0.322	8.11 v.s 3.6	0.004
Coherence	14.64 vs. 19.29	0.174	20.56 vs. 32	0.019	11.65 vs. 10.06	0.513	13.78 vs. 9.2	0.05
Other	0 vs. 0	N/A	12.22 vs.18.4	0.053	0 vs. 0	N/A	6.78 vs. 8.4	0.519

Table 6.7: T-test Results of *select* Justification Frequency w.r.t. Correctness (Individual vs. Collaboration).

6.3.3.1 Individual vs. Collaboration

Figure 6.6 and 6.7 present the frequency of justifications used for correct and incorrect responses, respectively. Table 6.7 provides detailed statistical significance scores for the comparison between individual and collaborative settings. The top three dominant categories mentioned by AMT participants for correct responses were 'grammar', 'common sense', and 'coherence', followed by 'lack of creativity'. On the other hand, the least frequent justifications used for correct responses were 'other', 'repetition', and 'self-contradiction'. Interestingly, no participants chose the 'other' option during the experiment to provide an explanation that did not fit into the 7 predefined categories.

Additionally, the usage of 'grammar, 'common sense', and 'coherence' categories increased the most from Individual to Collaboration settings, with 'grammar' and 'common sense' experiencing a significant increase of 9.11% and 6.23% respectively (p = 0.016, p = 0.036).

Moving on to the analysis of incorrect responses, we found that AMT participants frequently cited 'grammar', 'logical errors', 'self-contradiction', and 'lack of creativity' as justifications for their choices. Interestingly, the frequency of all these categories decreased from Individual to Collaboration, except for 'grammar'. Moreover, the drop in 'logical errors' and 'self-contradiction' was found to be statistically significant.

The Upwork participants commonly provided the following justifications - 'coherence', 'grammar', and 'common sense', regardless of whether they discussed the task or not. The 'other' category was also frequently chosen, with off-topic and off-prompt being the most common justifications provided. In contrast to AMT workers, Upwork participants rarely cited 'repetition' to justify their decisions. However, when they solved

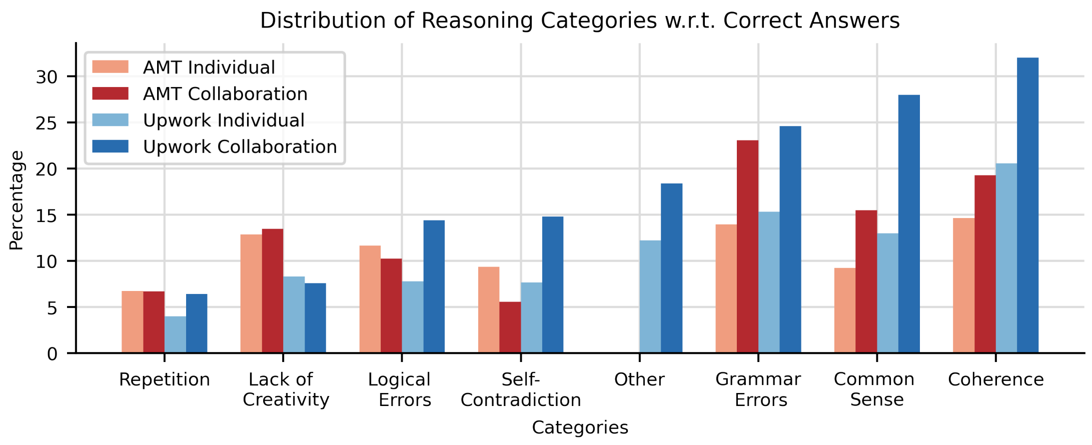


Figure 6.6: Justification Category Distribution w.r.t. Correct Responses. "Image from [3]."

the task collaboratively, there was a significant increase in the frequency of providing certain reasons. These included 'grammar' (+9.27%, p = 0.013), 'common sense' (+15%, p < 0.0001), 'logical errors' (+6.62%, p = 0.006), 'self-contradiction' (+7.13%, p = 0.011), and 'coherence' (+11.44%, p = 0.019). The only category that was used less frequently in collaborative problem-solving than individual problem-solving was 'Lack of Creativity', but it was not statistically significant.

Furthermore, the most commonly used justifications for incorrect answers from Upwork participants in the individual-based setting were 'grammar', 'coherence', and 'lack of creativity'. However, for team-based collaboration, all justification types except 'repetition', 'common sense', 'logical errors', and 'other' decreased in frequency. Despite an increase in the frequency of 'grammar', 'common sense', 'logical errors', and 'self-contradiction' categories, statistical significance was not observed. Moreover, during the team-based approach, 'lack of creativity' and 'coherence' categories were less commonly cited than in the individual-based approach.

6.3.3.2 Non-Experts vs. Experts

We present the comparative analysis results between AMT-Individual vs. Upwork-Individual and AMT-Collaboration vs. Upwork-Collaboration in Table 6.8. It is noteworthy that while AMT participants are non-experts, Upwork participants are experts.

The analysis of correct responses indicates that 'grammar' and 'coherence' were

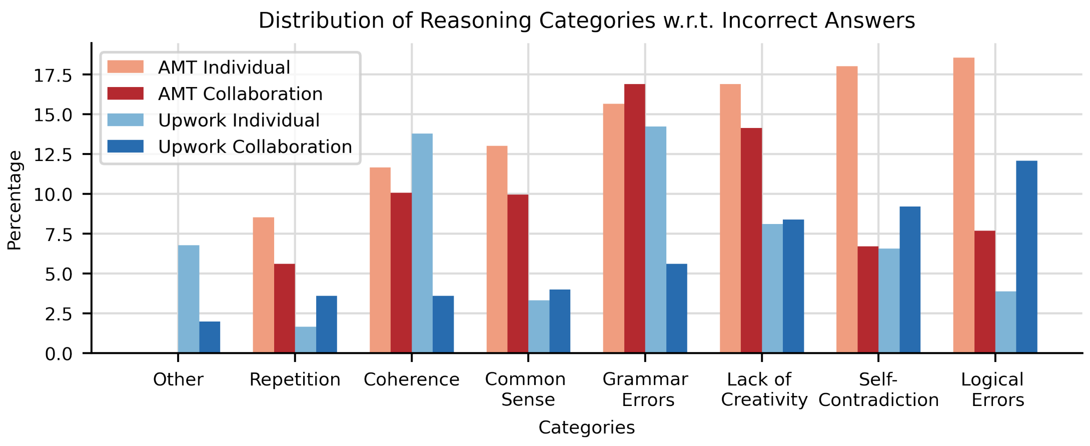


Figure 6.7: Justification Category Distribution w.r.t. Incorrect Responses. "Image from [3]."

Justification Type	Correct				Incorrect			
	Individual		Collaboration		Individual		Collaboration	
	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value
Grammar	13.98 vs. 15.33	0.57	23.08 vs. 24.6	0.747	15.65 vs. 14.22	0.53	16.89 vs. 12.07	0.175
Repetition	6.73 vs. 4	0.107	6.69 vs. 6.4	0.919	8.54 vs. 1.67	3.46e-07	5.62 vs. 2	0.029
Common Sense	9.25 vs. 13	0.086	15.48 vs. 28	0.004	13.02 vs. 3.33	6.49e-07	9.95 vs. 5.6	0.06
Logical Errors	11.64 vs. 7.77	0.056	10.24 vs. 14.4	0.151	18.54 vs. 3.89	1.14e-10	7.7 vs. 4	0.089
Self-Contradiction	9.35 vs. 7.67	0.398	5.57 vs. 14.8	0.002	18.01 vs. 6.56	4.7e-08	6.7 vs. 3.6	0.097
Lack of Creativity	12.87 vs. 8.33	0.026	13.49 vs. 7.6	0.071	16.9 vs. 8.11	1.17e-05	14.13 vs. 3.6	7.26e-05
Coherence	14.64 v.s 20.56	0.066	19.29 vs. 32	0.011	11.65 vs. 13.78	0.283	10.06 vs. 9.2	0.75
Other	0 vs. 12.22	1.1e-11	0 vs. 18.4	1.11e-09	0 vs. 6.78	6.5e-08	0 vs. 8.4	0.0003

Table 6.8: T-test Results of *select* Justification Frequency w.r.t. Correctness (Non-experts vs. Experts).

the most frequently cited justifications among individuals of different expertise levels, including both AMT and Upwork participants. Interestingly, 'repetition' was the least commonly used justification by both groups. Upwork participants tended to use 'grammar', 'common sense', 'coherence', and 'other' more often than AMT participants, although only the difference in the use of 'other' was statistically significant (0% vs. 12.22%, $p < 0.0001$). This is potentially because AMT participants did not have the opportunity to provide written explanations for their choices. Moreover, the frequency of 'lack of creativity' was significantly lower among Upwork-Individuals (8.33%) compared to AMT-Individuals (12.87%) ($p = 0.0264$).

Although 'grammar' and 'coherence' were still popular justifications for correct answers among both AMT and Upwork participants in the collaborative setting, Upwork par-

Justifications	Code
Writing is erratic (i.e., does not have a good flow)	Low
Grammatical issues	
Repetition	
Sentence structure issues	
Lacks common sense	High
Lack of creativity or boring to read	
Contains logical errors/fallacies	
Contradicts previous sentences	
Too much/too little information	
Off-prompt	
Off-topic	
Incorrect Information	
low-level+ high-level	Hybrid

Table 6.9: Code book for error level annotation.

ticipants cited ‘common sense’ more frequently than ‘grammar’ in Upwork-Collaboration. In addition, justifications such as ‘contradicts previous sentences’ (+9.23%, $p = 0.002$), ‘lacks common sense’ (+12.7%, $p = 0.011$), and ‘coherence’
(+12.51%, $p = 0.004$) was more strongly associated with correct responses in Upwork-Collaboration than in AMT-Collaboration.

AMT-Individuals and Upwork-Individuals cited ‘lack of creativity’ as a top-3 justification for incorrect responses, while AMT-Individuals also frequently used ‘logical errors’ and ‘self-contradiction’, and Upwork-Individuals often chose ‘grammar’ and ‘coherence’. Except for ‘coherence’ and ‘other’, all justification categories decreased in frequency from AMT-Individuals to Upwork-Individuals. However, the decrease in ‘grammar’ was not statistically significant ($p > 0.05$), and despite an increase in mentions of ‘coherence’, the difference was also not significant. Both AMT and Upwork participants mentioned grammatical errors the most within incorrect answers in the collaborative context. Interestingly, repetition’ (-3.63%, $p = 0.029$) and ‘lack of creativity’ (-10.53%, $p < 0.0001$) were the only categories that showed a significant drop in frequency from AMT-Collaboration to Upwork-Collaboration.

6.3.4 Justification Patterns - *write* (RQ3)

The *write* setting differs from the *select* setting in that it does not limit the error types to a predetermined set of 7 (excluding ‘other’). This allows for more in-depth analysis. Inspired by [191]’s discovery that participants tended to focus on the text’s form rather

Justification Level	Correct				Incorrect			
	AMT		Upwork		AMT		Upwork	
	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value
Low	19.54 vs. 21.08	0.675	21.89 vs. 20.4	0.668	24.95 vs. 19.78	0.121	23.11 vs. 12.8	0.0009
High	20.94 vs. 21.88	0.812	20.33 vs. 19.13	0.744	25.75 vs. 24	0.68	15.11 vs. 8.4	0.006
Hybrid	5.56 vs. 9.67	0.103	13.67 vs. 28.67	0.0003	3.27 vs. 3.6	0.825	5.67 vs. 9.6	0.067

Table 6.10: T-test Results of Justification Level Frequency w.r.t. Correctness (Individual v.s. Collaboration).

than its content, we manually reviewed and sorted the justifications into three categories: low-level, high-level, and hybrid. A low-level justification pertains to the text's format, style, and tone; a high-level justification identifies errors based on the text's meaning, and a hybrid justification indicates instances where evaluators used both low-level and high-level justifications. One researcher created a codebook (Table 6.9) with 7 predefined categories and initially coded responses provided by AMT workers. We added new justifications to the codebook as we encountered additional information in the data. Subsequently, two other researchers independently labeled the responses. The agreement among the three annotators was evaluated using Fleiss' Kappa coefficient [198] (0.924 for individual responses and 0.94 for collaborative responses), indicating high reliability in the generated labels.

As the Upwork experiments did not include the writing setting, we had to rely on the justifications provided in the *select* style, including those categorized as 'Other'. We then conducted T-tests to compare the distribution differences across the three labels.

6.3.4.1 Individual vs. Collaboration

Table 6.10 illustrates that when solving the task individually, AMT participants' correct responses were associated more strongly with either low or high error levels, rather than both. In the collaborative setting, their use of hybrid reasoning, as well as low and high-level justifications, increased, but these increases were not statistically significant. Regarding incorrect answers, approximately 25% of AMT individuals used low-level or high-level justifications to explain their judgments, while only 3% provided both. When performing the task collectively, the frequency of low-level justifications decreased to 20%, but the drop was not statistically significant. There was minimal change in high and hybrid-level errors.

Justification Level	Correct				Incorrect			
	Individual		Collaboration		Individual		Collaboration	
	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value	Mean Accuracy	p-value
Low	19.54 vs. 21.89	0.482	21.08 vs. 20.4	0.858	24.95 vs. 23.11	0.558	19.78 vs. 12.8	0.032
High	20.94 vs. 20.33	0.838	21.88 vs. 19.13	0.545	25.75 vs. 15.11	0.003	24 vs. 8.4	8.9e-06
Hybrid	5.56 vs. 13.67	0.0007	9.67 vs. 28.67	1.3e-05	3.27 vs. 5.67	0.083	3.59 vs. 9.6	0.008

Table 6.11: T-test Results of Justification Level Frequency w.r.t. Correctness (Non-Experts v.s. Experts).

In individual problem-solving, Upwork participants, like their AMT counterparts, also cited one error type more frequently than both for correct responses. However, the frequency of hybrid reasoning almost doubled after group discussions, becoming the most common justification type with $p < 0.05$. There was no significant decrease in low or high-level errors, however. When it came to incorrect responses, low-level errors were the most frequently mentioned by Upwork participants in individual problem-solving, with an average of 23.11% per question using this justification type. In a collaborative environment, this percentage dropped to 12.8%, and the drop was statistically significant. Similarly, the shift in frequency of high-level justifications from 15.11% to 8.4% was significant. Hybrid-level errors, however, became more frequent after collaboration but were not significant.

6.3.4.2 Non-Experts vs. Experts

Table 6.11 presents the T-test outcomes for comparing the AMT and Upwork experiments. The results indicated that, in both individual and collaborative problem-solving settings, the proportion of participants who cited low- or high-level reasoning for their correct responses did not significantly vary between AMT and Upwork studies. Regardless of their language proficiency, the percentages of low- or high-level justifications ranged from 20% to 23%. However, for individual problem-solving, the frequency of hybrid reasoning was significantly higher in Upwork (8.11% more) than in AMT. In the collaborative environment, the gap between AMT and Upwork in the frequency of both low- and high-level justifications increased from 8.11% to 19%, and the difference became statistically more significant.

For incorrect responses, the difference between AMT and Upwork individual participants in citing low-level errors as their justification was negligible, with $p > 0.05$. The difference in hybrid-level errors was also not significant. However, there was a statistically significant difference in the frequency of high-level explanations, with Upwork participants having a much lower frequency than AMT participants.

In the collaborative setting, AMT participants cited more low- and high-level justifications compared to Upwork participants. Specifically, the rate of high-level justifications was approximately three times greater in the AMT setting. Conversely, hybrid-level errors were more common in Upwork, and the difference was significant.

6.4 Discussion

6.4.1 Summary of Results

This paper presents a comprehensive analysis of the performance of laypeople on Amazon Mechanical Turk (AMT) and English professionals on Upwork in detecting deepfake texts. Additionally, it explores the impact of asynchronous and synchronous collaboration on their performance and sheds light on the correlation between different textual elements and detection accuracy. The key findings from Section 6.3 are summarized below:

Individual vs. Collaboration

1. **Collaboration only significantly improved Experts' performance**: Although there is no significant difference in deepfake text detection performance between AMT-Individual and AMT-Collaboration, Upwork-Collaboration outperforms Upwork-Individual significantly, with a detection rate of 69% compared to 56%.

2. **In the collaborative setting, experts cited coherence, logical fallacies, and self-contradiction errors significantly more often as justifications for deepfake text detection compared to the individual setting**: There is a difference in the frequency of citing coherence, logical fallacies, and self-contradiction errors between Upwork-Individual and Upwork-Collaboration, with Collaboration citing these errors more frequently, while no difference was observed in the AMT experiments.

3. **Non-experts showed no significant difference in their use of the three error levels between the individual and collaborative scenarios, unlike the experts**: Although there were no differences in the frequency of low, high, and hybrid error levels between AMT-Individual and AMT-Collaboration, Upwork-Collaboration's

use of the hybrid level was significantly higher than Upwork-Individual - (Individual vs. Collaboration usage percentage - 13.67 vs. 28.67 for correct responses).

Experts vs. Non-experts

1. **Experts outperformed non-experts in terms of detection accuracy for deepfake texts**: Although both AMT (45%) and Upwork (56%) individual groups showed a significant improvement in their deepfake text detection performance compared to the baseline (32% and 33%, respectively), Upwork participants outperformed AMT participants in both individual and collaborative environments.

2. **Non-experts focused more on the creativity of articles than experts**: Upwork participants provided fewer justifications related to a lack of creativity compared to AMT participants for correct responses. When comparing non-experts to experts (12.87 vs. 8.33 for correct responses and 13.49 vs. 7.6 for incorrect responses), creative errors were used more frequently as a justification by non-experts.

3. **Experts showed a greater focus on consistency issues within the articles compared to non-experts**: Upwork participants were able to identify errors related to consistency, such as off-topic and off-prompt, which AMT participants were unable to detect.

4. **Experts tended to use hybrid-level justifications more frequently than non-experts.**: The use of hybrid-level justifications was more frequent among Upwork participants than AMT participants, particularly in the collaborative setting for correct responses. The comparison of non-experts versus experts in the collaborative setting showed a frequency of 9.67 versus 28.67 for correct responses in the use of hybrid-level errors.

6.4.2 Implications

We will examine the implications of the summarized results below to deduce the underlying reasoning or phenomena behind the findings.

6.4.2.1 Individual vs. Collaboration

1. **Collaboration only significantly improved Experts' performance**: It is widely acknowledged that a group's performance can surpass that of an individual's, even

the most knowledgeable person [199]. This phenomenon is further confirmed in the performance between individual and asynchronous collaboration for AMT experiments, while Upwork-Individual participants not only outperformed the baseline but also benefited from synchronous collaboration. One possible reason for this is that synchronous collaboration encouraged workers to be more engaged, creative, and sociable. Existing literature in the field of Computer-Supported Cooperative Work (CSCW) has argued that the advantages of synchronous collaboration outweighs those of asynchronous collaboration [200–202]. Another explanation for these mixed results could be that Upwork collaborators share the same English expertise background, which may have positively impacted their group intelligence due to their familiarity with each other's fields and individual high intelligence levels.

2. **In the collaborative setting, experts cited coherence, logical fallacies, and self-contradiction errors significantly more often as justifications for deepfake text detection compared to the individual setting**: The study found that non-experts did not show any differences in the use of coherence, logical errors, and self-contradiction justifications between individual and collaborative settings. However, experts used coherence and self-contradiction justifications more frequently in collaboration when detecting deepfake texts accurately and less frequently when detecting them inaccurately. This suggests that these high-level errors (Table 6.9) are challenging to detect and require advanced linguistic skills, which English professionals are more likely to possess. The results reported by [192] confirm this phenomenon. In contrast, logical fallacy errors were used more frequently by experts in the collaborative setting but were found to be a weaker predictor of deepfake texts.

3. **Non-experts showed no significant difference in their use of the three error levels between the individual and collaborative scenarios, unlike the experts**: We categorized errors into low, high, and hybrid levels based on the form and content of the text. Non-experts did not show any significant pattern in error level usage between independent and collaborative settings for both correct and incorrect responses, suggesting that their collaboration did not improve their performance significantly. In contrast, experts showed a more than doubled prevalence of hybrid-level errors for correct responses when collaborating, indicating that discussing

and finding more errors were possible in collaboration than working independently. The use of hybrid-level errors suggests that considering both form and content-wise error is a useful strategy for deepfake text detection.

6.4.2.2 Experts vs. Non-experts

1. **Experts outperformed non-experts in terms of detection accuracy for deepfake texts**: Unlike previous studies where humans performed at or below the chance level [1, 117, 118, 191], both experts and non-experts in our study surpassed the baseline performance. We tried various interface designs and example-based training to improve the performance of non-experts. However, Upwork participants with profound writing experience performed significantly better than AMT workers. Interestingly, non-experts who received additional training was still unable to outperform experts without any training, suggesting that existing training procedures are insufficient for non- experts. Therefore, more nuanced approaches tailored to the patterns that English professionals adopt should be explored.

2. **Non-experts focused more on the creativity of articles than experts**: Experts mentioned a lack of creativity less frequently than non-experts for correct responses, but their superior performance suggests that lack of creativity is not a strong indicator of deepfake texts. Non-experts frequently mentioned lack of creativity for both correct and incorrect responses, reinforcing this argument. The Upwork participants were able to leverage their knowledge of writing domains and expected writing styles in political news articles when making decisions. Therefore, creativity errors may be a misleading indicator for detecting deepfake texts in the politics-related news domain and further research is needed for other news topics.

3. **Experts showed a greater focus on consistency issues within the articles compared to non-experts**: Expert participants were more successful than non- experts in detecting off-topic and off-prompt errors, which are challenging for language models. This is because these types of errors require careful reading and thinking. Additionally, experts' ability to detect errors that go beyond the provided error categories indicate that they possess advanced linguistic skills. This ability to identify consistency errors suggest that they are strong indicators of deepfake texts.

4. **Experts tended to use hybrid-level justifications more frequently than non-experts.**: Upon analyzing both non-experts and experts, we found a significant difference in the frequency of hybrid-level error detection. This suggests that a combination of low-level and high-level error detection is more effective in accurately identifying deepfake texts than relying on only one type of error. Detecting hybrid-level errors requires a more thorough analysis than identifying basic errors like grammar and repetition. While detecting low-level errors may be easier, relying solely on them can lead to erroneous judgments. Experts' ability to detect hybrid-level errors demonstrates their expertise in deepfake detection tasks, as they can look beyond obvious error types. The results also suggest that experts' deductive reasoning skills, advanced linguistic abilities, and prior experiences inform their use of hybrid-level errors.

6.5 Limitations

To implement design choices and run manageable experiments, we made a few simplifications that may limit our findings. First, since, we only use GPT-2 to generate deepfake texts, our findings may not be directly applicable to other NTGs. However, we believe that the choice of GPT-2 is reasonable because: (1) prior research reported that human detection performance of deepfake texts by the later GPT-3 and GPT-2 is similar [1, 191], and (2) using the largest parameter size of GPT-2 enabled us to generate deepfake texts more effectively that closely resembles GPT-3 quality. Furthermore, as we use the default hyperparameters of GPT-2 to generate the texts, we believe that the results may be limited to that sampling technique. However, we mitigated this issue by manually checking the quality of a few of the articles and found the deepfake texts to be human-like. This preserved the integrity of the experiments as the task remained non-trivial.

Next, for the *non-expert vs. expert* analysis, we compared AMT to Upwork, where AMT are the non-experts and Upwork, experts. However, since they are different forms of collaboration, AMT being asynchronous and Upwork being synchronous, the comparison may be different. Although a few prior works explored the space of synchronous collaboration between AMT workers, these systems tend to be engineering-heavy and are rarely used in real-world applications. We thus believe asynchronous collaboration is a more realistic way of using AMT. We understand that this design decision might compromise the comparability of the two settings but we believe it is a reasonable trade-off. Furthermore, Upwork's interface supports the recruitment of English experts, as well as provides a framework for synchronous collaboration. To mitigate this limitation, we calculate the accuracy of AMT and Upwork participants in the same way (per-article accuracy) to have a fairer comparison.

6.6 Summary

In this paper, we studied human performance in deepfake text detection. To be more realistic, we built a 3-paragraph article with 1/3 paragraphs, machine-generated (deepfake) and 2/3 paragraphs, human-written. We ask human participants to select which paragraph is deepfake and to provide justification for their selection out of 7 error types. Specifically, we studied human performance with two variables - *individual vs. collaboration* and *English non-expert vs. English expert*. To achieve this, we recruit non-expert human participants from AMT and experts from Upwork. Furthermore, we run asynchronous collaboration with AMT and compared it to synchronous collaboration with Upwork. Finally, our results suggest that the synchronous collaboration of expert human participants significantly improves human performance in deepfake text detection. We further identify several factors (such as coherence and consistency) that deepfake texts excel at or fail by analyzing their justification patterns. Lastly, the enhanced performance of participants (particularly non-experts) from baseline in the individual setting indicates that our *Turing Test* framework facilitates the improvement of humans' deepfake text detection performance.

Chapter 7 | Open Challenges and Conclusion

7.1 Open Challenges in Deepfake Text Detection

Although there have been several meaningful works on the current landscape of Authorship Attribution (AA) and Authorship Obfuscation (AO) models, the two research problems are still in their early development, especially the direction of *deepfake text detection*. In this section, as such, we discuss some of the remaining challenges. Also, see Figure 7.1 for a Venn diagram of AA & AO open problems and challenges for deepfake text detection.

7.1.1 Need for Comprehensive Benchmark

Generally, existing literature tends to create or use particular datasets in silos, making their findings limited and incomparable across the literature. As mentioned in [19, 30], however, the study of DTD can be greatly improved with the availability of more comprehensive and generalizable datasets whose coverage varies across diverse: (1) domains (e.g., news, online forum, recipe, stories), (2) language models, (3) decoding strategies, or (4) length of texts. Further, not all AA/AO models share their codebase and experimental configurations, making the comparative analysis difficult. However, generating and maintaining a large number of deepfake texts across different settings cost significant resources and effort. [1] attempted to propose a benchmark for AA research, *TuringBench*, but it does not satisfy the needs fully. Therefore, it is greatly needed to develop a comprehensive benchmark with diverse datasets of AA/AO problems,

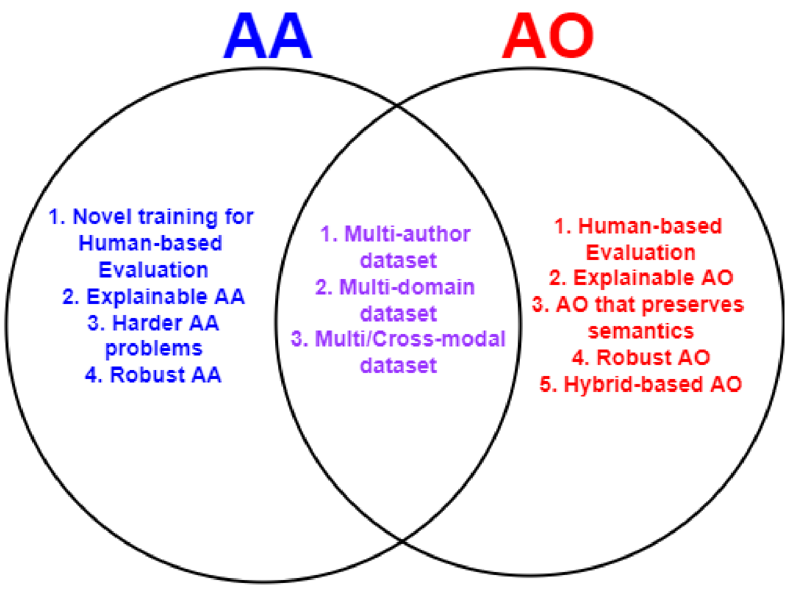


Figure 7.1: Current Open problems and challenges in the Authorship Attribution (AA) and Authorship Obfuscation (AO) for deepfake text detection.

along with the codebase of known methods in a unified environment, so that objective comparison can be carefully performed to understand the pros and cons of existing solutions and brainstorm new ideas for improvement.

7.1.2 Call for Complex AA/AO Variations

With the introduction of "machine" in writing high-quality texts, the set-up of "authors" in future scenarios can be more complex. For instance, one could generate a more realistic text using multiple language models in sequence (e.g., each language model improves upon the text generated by another language model in a previous step) or in parallel (e.g., each language model generates only parts of a long text). Symmetrically, it is also plausible to use multiple AO solutions in sequence or parallel to improve the overall performance of obfuscation. Yet another possible scenario is to think of "human-in-the-loop" attribution or obfuscation. For instance, would a team (of humans, of machines, or of humans and machines) outperform an individual (of human or machine) in solving AA or AO task? To our best knowledge, there is no study of AA/AO for such complex scenarios.

7.1.3 Need for Interpretable AA/AO Models

Currently, there are only a few interpretable AA models (e.g., GLTR) and AO techniques (e.g., Homoglyph) for deepfake texts, as summarized in Tables **??** and 2.4, respectively. That is, when an AA model detects a text as machine-generated or human-written, or when an AO model modifies parts of a text to hide authorship, it often cannot explain "why?" Ideally, however, such models should be able to provide an easy-to-understand and intuitive explanation, especially to users with no linguistic expertise, as to why a given text is attributable to a particular DTG or why a particular phrase of a text is critical to reveal an author's identity. In addition, more research is needed to develop an intuitive human interface or visualization toward explainable AA/AO models.

7.1.4 Need for Improved Human Training

In parallel to improving the performance of AA/AO solutions, it is equally important to raise the awareness of AA/AO problems in the presence of deepfake texts, and to be able to train human users to detect deepfake texts better (e.g., identify phishing or misinformation message that includes deepfake texts as parts) or use AO solutions to hide one's authorship (e.g., an activist posting his/her message on social media without revealing true identity). As we illustrate in Section 2.5.2, however, humans are not good at detecting deepfake texts, and there are not many AA/AO solutions suitable for novice users to benefit from in solving AA/AO tasks. Worst, still, is that even a few AA models such as *GLTR* that were shown to be able to help human users to detect deepfake texts better have become less effective with the advancement of DTGs. Therefore, great needs exist to have a better way to train human users in solving AA/AO tasks.

7.1.5 Call for Robust AA/AO Solutions

In section 2.4, we surveyed all literature that evaluated the robustness of AA/AO models and found that most existing AO techniques do not preserve the original semantics of text well and thus cannot easily evade the attribution of AA solutions, especially human detection. Similarly, as we adopt more sophisticated hybrid approaches for AA tasks, successful AO attacks to hide authorship will become more challenging. Part of the reason for these vulnerabilities in existing AA/AO solutions is that the bulk of existing literature has studied either AA or AO problems in separation, thus greatly limiting their robustness against the other problem. Therefore, to stay relevant and synonymous with a real-life scenario, both AA and AO solutions need to learn from each other, and co-train/co-evolve, as in a min-max optimization game.

7.2 Conclusion

In this dissertation, we answer 3 Research Questions (RQs) - (1) Can one detect subtle linguistic differences between deepfake texts and human-written ones? (2) Can one build an accurate deepfake text detector to distinguish between human vs. deepfake text authorship? (3) How can one improve human performance in detecting deepfake texts?

For RQ1, we investigated, distinguishing linguistic differences between deepfake texts and human-written texts. To answer this RQ, we use LIWC (Linguistic Inquiry & Word Count), Flesch's readability measure and entropy (number of unique characters) features to examine these texts. We find that humans write more formally in the context of news articles and less revealing in writing.

Next, for RQ2, which is the main part of this dissertation, we implemented several types of automatic deepfake text detectors. First, we implemented a stylometric detector using the linguistic features extracted for RQ1. This yielded a superior performance of 90% macro F1 score in attributing the authorships of 1 human vs. 8 LLMs. Second, we attribute the authorship of 1 human vs. 19 LLMs using 2 deep learning-based detectors - BERT & RoBERTa. This detector entails fine-tuning BERT and RoBERTa base models on our proposed benchmark dataset - TuringBench. Finally, we surveyed Authorship Attribution (AA) and Authorship Obfuscation (AO) techniques for deepfake texts and find that hybrid AA solutions perform better in terms of accuracy and robustness. Therefore, we propose an end-to-end hybrid-based deepfake text detector which is an ensemble of a Transformer-based model and Topological Data Analysis (TDA) techniques. We find that this technique is especially robust to noisy and imbalanced dataset which is the realistic nature of generalizable deepfake text datasets.

Lastly, for RQ3, the hypothesis here is that it is not enough to only build accurate automatic deepfake text detectors without understanding how humans perform in the task. To that end, we initially performed 2 studies - (1) given this one article, can you decide if it is machine-generated or not? (2) given 2 articles, one machine-generated and the other, human-written, can you decide which is machine-generated? The human participants underperformed on this task, achieving random-guessing accuracy. Interestingly, for study (1), we only provided the human participants with machine-generated texts, which they still failed to accurately identify. Thus, given the difficulty of this task, we proposed a novel solution for improving human performance through collaboration. Furthermore, we hypothesized that English experts will perform better on the task of deepfake text detection than English non- experts. English experts in this context are defined as people who have earned an educational degree in English or a related program, while non-experts are people without an educational degree in English or related programs. Therefore, we compare the performance of English non-experts and English experts in an individual and collaborative setting. We also compare asynchronous collaboration for non-experts and synchronous collaboration for experts. Intuitively, we find that collaboration improves performance, however, the improvement is only significant for experts.

Bibliography

[1] Uchendu, A., Z. Ma, T. Le, R. Zhang, and D. Lee (2021) "TURINGBENCH: A Benchmark Environment for Turing Test in the Age of Neural Text Generation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2001–2016.

[2] Gehrmann, S., H. Strobelt, and A. M. Rush (2019) "GLTR: Statistical Detection and Visualization of Generated Text," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 111–116.

[3] Uchendu, A., J. Lee, H. Shen, T. Le, T.-H. Huang, and D. Lee (2023) "Understanding Individual and Team-based Human Factors in Detecting Deepfake Texts," *arXiv preprint arXiv:2304.01002*.

[4] Turing, A. M. (2009) "Computing machinery and intelligence," in *Parsing the turing test*, Springer, pp. 23–65.

[5] Li, J., T. Tang, W. X. Zhao, and J.-R. Wen (2021) "Pretrained language models for text generation: A survey," *arXiv preprint arXiv:2105.10311*.

[6] Zhang, H., H. Song, S. Li, M. Zhou, and D. Song (2022) "A survey of controllable text generation using transformer-based pre-trained language models," *arXiv preprint arXiv:2201.05337*.

[7] Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever (2018) "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

[8] Ng, N., K. Yee, A. Baevski, M. Ott, M. Auli, and S. Edunov (2019) "Facebook FAIR's WMT19 News Translation Task Submission," *arXiv preprint arXiv:1907.06616*.

[9] CHEN, P.-J., A. LEE, C. WANG, N. GOYAL, A. FAN, M. WILLIAMSON, and J. GU (2020) "Facebook AI's WMT20 News Translation Task Submission," in *Proceedings of the Fifth Conference on Machine Translation*, pp. 113–125.

[10] KESKAR, N. S., B. MCCANN, L. R. VARSHNEY, C. XIONG, and R. SOCHER (2019) "Ctrl: A conditional transformer language model for controllable generation," *arXiv preprint arXiv:1909.05858*.

[11] DATHATHRI, S., A. MADOTTO, J. LAN, J. HUNG, E. FRANK, P. MOLINO, J. YOSINSKI, and R. LIU (2020) "Plug and Play Language Models: A Simple Approach to Controlled Text Generation," in *International Conference on Learning Representations*.

[12] RAFFEL, C., N. SHAZEER, A. ROBERTS, K. LEE, S. NARANG, M. MATENA, Y. ZHOU, W. LI, and P. J. LIU (2020) "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, **21**(1), pp. 5485–5551.

[13] WOLF, T., L. DEBUT, V. SANH, J. CHAUMOND, C. DELANGUE, A. MOI, P. CISTAC, T. RAULT, R. LOUF, M. FUNTOWICZ, ET AL. (2019) "HuggingFace's Transformers: State-of-the-art Natural Language Processing," *ArXiv*, pp. arXiv–1910.

[14] KREPS, S., R. M. MCCAIN, and M. BRUNDAGE (2022) "All the news that's fit to fabricate: AI-generated text as a tool of media misinformation," *Journal of Experimental Political Science*, **9**(1), pp. 104–117.

[15] UCHENDU, A., T. LE, K. SHU, and D. LEE (2020) "Authorship attribution for neural text generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8384–8395.

[16] ZELLERS, R., A. HOLTZMAN, H. RASHKIN, Y. BISK, A. FARHADI, F. ROESNER, and Y. CHOI (2019) "Defending against neural fake news," in *Advances in Neural Information Processing Systems*, pp. 9051–9062.

[17] BAKHTIN, A., S. GROSS, M. OTT, Y. DENG, M. RANZATO, and A. SZLAM (2019) "Real or fake? learning to discriminate machine from human generated text," *arXiv preprint arXiv:1906.03351*.

[18] SCHUSTER, T., R. SCHUSTER, D. J. SHAH, and R. BARZILAY (2020) "The Limitations of Stylometry for Detecting Machine-Generated Fake News," *Computational Linguistics*, **46**(2), pp. 499–510.

[19] FRÖHLING, L. and A. ZUBIAGA (2021) "Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover," *PeerJ Computer Science*, **7**, p. e443.

[20] Pillutla, K., S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui (2021) "An information divergence measure between neural text and human text," *arXiv preprint arXiv:2102.01454*.

[21] Gallé, M., J. Rozen, G. Kruszewski, and H. Elsahar (2021) "Unsupervised and Distributional Detection of Machine-Generated Text," *arXiv preprint arXiv:2111.02878*.

[22] Abbasi, A. 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), pp. 1–29.

[23] Tan, R., B. Plummer, and K. Saenko (2020) "Detecting Cross-Modal Inconsistency to Defend Against Neural Fake News," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2081–2106.

[24] Kushnareva, L., D. Cherniavskii, V. Mikhailov, E. Artemova, S. Barannikov, A. Bernstein, I. Piontkovskaya, D. Piontkovski, and E. Burnaev (2021) "Artificial Text Detection via Examining the Topology of Attention Maps," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 635–649.

[25] Stiff, H. and F. Johansson (2022) "Detecting computer-generated disinformation," *International Journal of Data Science and Analytics*, **13**(4), pp. 363–383.

[26] Uchendu, A., T. Le, K. Shu, and D. Lee (2020) "Authorship Attribution for Neural Text Generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[27] Zhong, W., D. Tang, Z. Xu, R. Wang, N. Duan, M. Zhou, J. Wang, and J. Yin (2020) "Neural Deepfake Detection with Factual Structure of Text," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2461–2470.

[28] Orzhenovskii, M. (2022) "Detecting Auto-generated Texts with Language Model and Attacking the Detector," *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference Dialogue 2022*.

[29] Diwan, N., T. Chakraborty, and Z. Shafiq (2021) "Fingerprinting Fine-tuned Language Models in the Wild," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4652–4664.

[30] Munir, S., B. Batool, Z. Shafiq, P. Srinivasan, and F. Zaffar (2021) "Through the looking glass: Learning to attribute synthetic text generated by

language models," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1811–1822.

[31] BROWN, T., B. MANN, N. RYDER, M. SUBBIAH, J. D. KAPLAN, P. DHARIWAL, A. NEELAKANTAN, P. SHYAM, G. SASTRY, A. ASKELL, S. AGARWAL, A. HERBERT-VOSS, G. KRUEGER, T. HENIGHAN, R. CHILD, A. RAMESH, D. ZIEGLER, J. WU, C. WINTER, C. HESSE, M. CHEN, E. SIGLER, M. LITWIN, S. GRAY, B. CHESS, J. CLARK, C. BERNER, S. MCCANDLISH, A. RADFORD, I. SUTSKEVER, and D. AMODEI (2020) "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, Curran Associates, Inc., pp. 1877–1901.
URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`

[32] GUERRERO, J. and I. ALSMADI (2022) "Synthetic Text Detection: Systemic Literature Review," *arXiv preprint arXiv:2210.06336*.

[33] IPPOLITO, D., D. DUCKWORTH, C. CALLISON-BURCH, and D. ECK (2020) "Automatic Detection of Generated Text is Easiest when Humans are Fooled," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1808–1822.

[34] CLARK, E., T. AUGUST, S. SERRANO, N. HADUONG, S. GURURANGAN, and N. A. SMITH (2021) "All That's 'Human'Is Not Gold: Evaluating Human Evaluation of Generated Text," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7282–7296.

[35] BUCHANAN, B., A. LOHN, M. MUSSER, and K. SEDOVA (2021) "Truth, lies, and automation," *Center for Security and Emerging Technology*, **1**(1), p. 2.

[36] ADELANI, D. I., H. MAI, F. FANG, H. H. NGUYEN, J. YAMAGISHI, and I. ECHIZEN (2020) "Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection," in *International Conference on Advanced Information Networking and Applications*, Springer, pp. 1341–1354.

[37] VAROL, O., E. FERRARA, C. DAVIS, F. MENCZER, and A. FLAMMINI (2017) "Online human-bot interactions: Detection, estimation, and characterization," in *Proceedings of the international AAAI conference on web and social media*, vol. 11, pp. 280–289.

[38] RADFORD, A., J. WU, R. CHILD, D. LUAN, D. AMODEI, and I. SUTSKEVER (2019) "Language models are unsupervised multitask learners," *OpenAI blog*, **1**(8), p. 9.

[39] CONNEAU, A. and G. LAMPLE (2019) "Cross-lingual language model pretraining," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 7059–7069.

[40] YANG, Z., Z. DAI, Y. YANG, J. CARBONELL, R. R. SALAKHUTDINOV, and Q. V. LE (2019) "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, pp. 5754–5764.

[41] DAI, Z., Z. YANG, Y. YANG, J. G. CARBONELL, Q. LE, and R. SALAKHUTDINOV (2019) "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988.

[42] FEDUS, W., B. ZOPH, and N. SHAZEER (2022) "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," *Journal of Machine Learning Research*, **23**(120), pp. 1–39.

[43] GAO, L., S. BIDERMAN, S. BLACK, L. GOLDING, T. HOPPE, C. FOSTER, J. PHANG, H. HE, A. THITE, N. NABESHIMA, ET AL. (2020) "The Pile: An 800GB Dataset of Diverse Text for Language Modeling," *arXiv preprint arXiv:2101.00027*.

[44] BLACK, S., S. BIDERMAN, E. HALLAHAN, Q. ANTHONY, L. GAO, L. GOLDING, H. HE, C. LEAHY, K. MCDONELL, J. PHANG, ET AL. (2022) "GPT-NeoX-20B: An Open-Source Autoregressive Language Model," in *Proceedings of BigScience Episode\# 5–Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136.

[45] WANG, B. and A. KOMATSUZAKI (2021), "GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model," `https://github.com/kingoflolz/mesh-transformer-jax`.

[46] WANG, B. (2021), "Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX," `https://github.com/kingoflolz/mesh-transformer-jax`.

[47] LEWIS, M., Y. LIU, N. GOYAL, M. GHAZVININEJAD, A. MOHAMED, O. LEVY, V. STOYANOV, and L. ZETTLEMOYER (2020) "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880.

[48] CHOWDHERY, A., S. NARANG, J. DEVLIN, M. BOSMA, G. MISHRA, A. ROBERTS, P. BARHAM, H. W. CHUNG, C. SUTTON, S. GEHRMANN,

ET AL. (2022) “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*.

[49] BARHAM, P., A. CHOWDHERY, J. DEAN, S. GHEMAWAT, S. HAND, D. HURT, M. ISARD, H. LIM, R. PANG, S. ROY, ET AL. (2022) “Pathways: Asynchronous distributed dataflow for ML,” *Proceedings of Machine Learning and Systems*, **4**, pp. 430–449.

[50] ZHANG, S., S. ROLLER, N. GOYAL, M. ARTETXE, M. CHEN, S. CHEN, C. DEWAN, M. DIAB, X. LI, X. V. LIN, ET AL. (2022) “Opt: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*.

[51] KRAUSE, B., A. D. GOTMARE, B. MCCANN, N. S. KESKAR, S. JOTY, R. SOCHER, and N. F. RAJANI (2021) “GeDi: Generative Discriminator Guided Sequence Generation,” in *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4929–4952.

[52] HOLTZMAN, A., J. BUYS, L. DU, M. FORBES, and Y. CHOI (2019) “The curious case of neural text degeneration,” *arXiv preprint arXiv:1904.09751*.

[53] WOLFF, M. and S. WOLFF (2020) “Attacking neural text detectors,” *arXiv preprint arXiv:2002.11768*.

[54] CROTHERS, E., N. JAPKOWICZ, H. VIKTOR, and P. BRANCO (2022) “Adversarial Robustness of Neural-Statistical Features in Detection of Generative Transformers,” *arXiv preprint arXiv:2203.07983*.

[55] PU, J., Z. SARWAR, S. M. ABDULLAH, A. REHMAN, Y. KIM, P. BHATTACHARYA, M. JAVED, B. VISWANATH, V. TECH, and L. PAKISTAN (2023) “Deepfake Text Detection: Limitations and Opportunities,” *44th IEEE Symposium on Security and Privacy*.

[56] BHAT, M. M. and S. PARTHASARATHY (2020) “How Effectively Can Machines Defend Against Machine-Generated Fake News? An Empirical Study,” in *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pp. 48–53.

[57] PU, J., Z. HUANG, Y. XI, G. CHEN, W. CHEN, and R. ZHANG (2022) “Unraveling the Mystery of Artifacts in Machine Generated Text,” , pp. 6889–6898.

[58] AI, B., Y. WANG, Y. TAN, and T. SAMSON (2022) “Whodunit? Learning to Contrast for Authorship Attribution,” *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*.

[59] XING, E., T. LE, and D. LEE (2023) "ALISON: Fast Stylometric Authorship Obfuscation," *Technical Report*.

[60] LIYANAGE, V., D. BUSCALDI, and A. NAZARENKO (2022) "A Benchmark Corpus for the Detection of Automatically Generated Text in Academic Publications," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 4692–4700.

[61] CUTLER, J., L. DUGAN, S. HAVALDAR, and A. STEIN (2021) "Automatic Detection of Hybrid Human-Machine Text Boundaries," .

[62] BITEN, A. F., L. GOMEZ, M. RUSINOL, and D. KARATZAS (2019) "Good news, everyone! context driven entity-aware captioning for news images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12466–12475.

[63] ROSATI, D. (2022) "SynSciPass: detecting appropriate uses of scientific text generation," in *Proceedings of the Third Workshop on Scholarly Document Processing*, pp. 214–222.

[64] FAGNI, T., F. FALCHI, M. GAMBINI, A. MARTELLA, and M. TESCONI (2021) "TweepFake: About detecting deepfake tweets," *Plos one*, **16**(5), p. e0251415.

[65] HUGGINGFACE (2019) "GPT-2 Output Detector Demo," *https://huggingface.co/openai-detector/*.

[66] JAWAHAR, G., M. ABDUL-MAGEED, and L. LAKSHMANAN (2022) "Automatic Detection of Entity-Manipulated Text using Factual Knowledge," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 86–93.

[67] GAMBINI, M., T. FAGNI, F. FALCHI, and M. TESCONI (2022) "On pushing DeepFake Tweet Detection capabilities to the limits," in *14th ACM Web Science Conference 2022*, pp. 154–163.

[68] LIU, X., Z. ZHANG, Y. WANG, Y. LAN, and C. SHEN (2022) "CoCo: Coherence-Enhanced Machine-Generated Text Detection Under Data Limitation With Contrastive Learning," *arXiv preprint arXiv:2212.10341*.

[69] MITCHELL, E., Y. LEE, A. KHAZATSKY, C. D. MANNING, and C. FINN (2023) "Detectgpt: Zero-shot machine-generated text detection using probability curvature," *arXiv preprint arXiv:2301.11305*.

[70] KESTEMONT, M., E. MANJAVACAS, I. MARKOV, J. BEVENDORFF, M. WIEGMANN, E. STAMATATOS, B. STEIN, and M. POTTHAST (2021) "Overview of the cross-domain authorship verification task at PAN 2021," in *CLEF (Working Notes)*.

[71] STAMATATOS, E. (2016) "Authorship verification: a review of recent advances," *Research in Computing Science*, **123**, pp. 9–25.

[72] STAMATATOS, E., M. KESTEMONT, K. KREDENS, P. PEZIK, A. HEINI, J. BEVENDORFF, M. POTTHAST, and B. STEIN (2022) "Overview of the authorship verification task at PAN 2022," *Working Notes of CLEF*.

[73] BEVENDORFF, J., B. CHULVI, E. FERSINI, A. HEINI, M. KESTEMONT, K. KREDENS, M. MAYERL, R. ORTEGA-BUENO, P. PĘZIK, M. POTTHAST, ET AL. (2022) "Overview of PAN 2022: Authorship Verification, Profiling Irony and Stereotype Spreaders, Style Change Detection, and Trigger Detection," in *European Conference on Information Retrieval*, Springer, pp. 331–338.

[74] TYO, J., B. DHINGRA, and Z. C. LIPTON (2022) "On the State of the Art in Authorship Attribution and Authorship Verification," *arXiv preprint arXiv:2209.06869*.

[75] GEHMAN, S., S. GURURANGAN, M. SAP, Y. CHOI, and N. A. SMITH (2020) "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3356–3369.

[76] PENNEBAKER, J. W., M. E. FRANCIS, and R. J. BOOTH (2001) "Linguistic inquiry and word count: LIWC 2001," *Mahway: Lawrence Erlbaum Associates*, **71**(2001), p. 2001.

[77] FAST, E., B. CHEN, and M. S. BERNSTEIN (2016) "Empath: Understanding topic signals in large-scale text," in *Proceedings of the 2016 CHI conference on human factors in computing systems*, pp. 4647–4657.

[78] LAGUTINA, K., N. LAGUTINA, E. BOYCHUK, I. VORONTSOVA, E. SHLIAKHTINA, O. BELYAEVA, I. PARAMONOV, and P. DEMIDOV (2019) "A survey on stylometric text features," in *2019 25th Conference of Open Innovations Association (FRUCT)*, IEEE, pp. 184–195.

[79] KAUR, R., S. SINGH, and H. KUMAR (2019) "Authorship analysis of online social media content," in *Proceedings of 2nd International Conference on Communication, Computing and Networking*, Springer, pp. 539–549.

[80] JONES, K., J. R. NURSE, and S. LI (2022) "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*, vol. 16, pp. 429–440.

[81] PENNINGTON, J., R. SOCHER, and C. D. MANNING (2014) "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

[82] DEVLIN, J., M.-W. CHANG, K. LEE, and K. TOUTANOVA (2019) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.

[83] ZHANG, R., Z. HU, H. GUO, and Y. MAO (2018) "Syntax encoding with application in authorship attribution," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2742–2753.

[84] SHRESTHA, P., S. SIERRA, F. GONZALEZ, M. MONTES, P. ROSSO, and T. SOLORIO (2017) "Convolutional Neural Networks for Authorship Attribution of Short Texts," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

[85] JAFARIAKINABAD, F., S. TARNPRADAB, and K. A. HUA (2019) "Syntactic Recurrent Neural Network for Authorship Attribution," *arXiv preprint arXiv:1902.09723*.

[86] KRAUSE, B., L. LU, I. MURRAY, and S. RENALS (2016) "Multiplicative LSTM for sequence modelling," *arXiv preprint arXiv:1609.07959*.

[87] KRAUSE, B., I. MURRAY, S. RENALS, and L. LIANG (2017) "Multiplicative LSTM for sequence modelling," in *5th International Conference on Learning Representations*, pp. 2872–2880.

[88] LECUN, Y., S. CHOPRA, R. HADSELL, M. RANZATO, and F. HUANG (2006) "A tutorial on energy-based learning," *Predicting structured data*, **1**(0).

[89] GAGIANO, R., M. M.-H. KIM, X. J. ZHANG, and J. BIGGS (2021) "Robustness analysis of grover for machine-generated news detection," in *Proceedings of the The 19th Annual Workshop of the Australasian Language Technology Association*, pp. 119–127.

[90] GAO, C. A., F. M. HOWARD, N. S. MARKOV, E. C. DYER, S. RAMESH, Y. LUO, and A. T. PEARSON (2022) "Comparing scientific abstracts generated by ChatGPT to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers," *bioRxiv*, pp. 2022–12.

[91] CLARK, K., M.-T. LUONG, Q. V. LE, and C. D. MANNING (2020) "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*.

[92] LAN, Z., M. CHEN, S. GOODMAN, K. GIMPEL, P. SHARMA, and R. SORICUT (2019) "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*.

[93] PAGNONI, A., M. GRACIARENA, and Y. TSVETKOV (2022) "Threat Scenarios and Best Practices to Detect Neural Fake News," in *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 1233–1249.

[94] GAO, T., X. YAO, and D. CHEN (2021) "SimCSE: Simple Contrastive Learning of Sentence Embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910.

[95] HE, P., X. LIU, J. GAO, and W. CHEN (2021) "DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION," in *International Conference on Learning Representations*.
URL `https://openreview.net/forum?id=XPZIaotutsD`

[96] HE, P., J. GAO, and W. CHEN (2021) "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing," *arXiv preprint arXiv:2111.09543*.

[97] KIPF, T. N. and M. WELLING (2016) "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*.

[98] SHAMARDINA, T., V. MIKHAILOV, D. CHERNIANSKII, A. FENOGENOVA, M. SAIDOV, A. VALEEVA, T. SHAVRINA, I. SMUROV, E. TUTUBALINA, and E. ARTEMOVA (2022) "Findings of the The RuATD Shared Task 2022 on Artificial Text Detection in Russian," *arXiv preprint arXiv:2206.01583*.

[99] VARSHNEY, L. R., N. S. KESKAR, and R. SOCHER (2020) "Limits of detecting text generated by large-scale language models," in *2020 Information Theory and Applications Workshop (ITA)*, IEEE, pp. 1–5.

[100] MAHMOOD, A., F. AHMAD, Z. SHAFIQ, P. SRINIVASAN, and F. ZAFFAR (2019) "A Girl Has No Name: Automated Authorship Obfuscation using Mutant-X." *Proc. Priv. Enhancing Technol.*, **2019**(4), pp. 54–71.

[101] ZHANG, W. E., Q. Z. SHENG, A. ALHAZMI, and C. LI (2020) "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, **11**(3), pp. 1–41.

[102] KARADZHOV, G., T. MIHAYLOVA, Y. KIPROV, G. GEORGIEV, I. KOYCHEV, and P. 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*, Springer, pp. 173–185.

[103] MAHMOOD, A., Z. SHAFIQ, and P. SRINIVASAN (2020) "A Girl Has A Name: Detecting Authorship Obfuscation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2235–2245.

[104] ZHAI, W., J. RUSERT, Z. SHAFIQ, and P. SRINIVASAN (2022) "A Girl Has A Name, And It's... Adversarial Authorship Attribution for Deobfuscation," *arXiv preprint arXiv:2203.11849*.

[105] MCDONALD, A. W., 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, pp. 299–318.

[106] HAROON, M., F. ZAFFAR, P. SRINIVASAN, and Z. SHAFIQ (2021) "Avengers Ensemble! Improving Transferability of Authorship Obfuscation," *arXiv preprint arXiv:2109.07028*.

[107] TABATABAEI, M., J. HAKANEN, M. HARTIKAINEN, K. MIETTINEN, and K. SINDHYA (2015) "A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods," *Structural and Multidisciplinary Optimization*, **52**(1), pp. 1–25.

[108] GAO, J., J. LANCHANTIN, M. L. SOFFA, and Y. QI (2018) "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, pp. 50–56.

[109] NGUYEN-SON, H.-Q., N.-D. T. TIEU, H. H. NGUYEN, J. YAMAGISHI, and I. E. ZEN (2017) "Identifying computer-generated text using statistical analysis," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, pp. 1504–1511.

[110] LIU, Y., M. OTT, N. GOYAL, J. DU, M. JOSHI, D. CHEN, O. LEVY, M. LEWIS, L. ZETTLEMOYER, and V. STOYANOV (2019) "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*.

[111] JIN, D., Z. JIN, J. T. ZHOU, and P. SZOLOVITS (2020) "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 8018–8025.

[112] FAN, A., M. LEWIS, and Y. DAUPHIN (2018) "Hierarchical Neural Story Generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898.

[113] ZHU, W. and S. BHAT (2020) "GRUEN for Evaluating Linguistic Quality of Generated Text," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 94–108.

[114] BANERJEE, S. and A. LAVIE (2005) "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

[115] CER, D., Y. YANG, S.-Y. KONG, N. HUA, N. LIMTIACO, R. S. JOHN, N. CONSTANT, M. GUAJARDO-CESPEDES, S. YUAN, C. TAR, ET AL. (2018) "Universal sentence encoder for English," in *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pp. 169–174.

[116] DONAHUE, C., M. LEE, and P. LIANG (2020) "Enabling Language Models to Fill in the Blanks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2492–2501.

[117] DUGAN, L., D. IPPOLITO, A. KIRUBARAJAN, and C. CALLISON-BURCH (2020) "RoFT: A Tool for Evaluating Human Detection of Machine-Generated Text," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 189–196.

[118] DOU, Y., M. FORBES, R. KONCEL-KEDZIORSKI, N. A. SMITH, and Y. CHOI (2021) "Scarecrow: A framework for scrutinizing machine text," *arXiv preprint arXiv:2107.01294*.

[119] MIRSKY, Y. and W. LEE (2021) "The creation and detection of deepfakes: A survey," *ACM Computing Surveys (CSUR)*, **54**(1), pp. 1–41.

[120] UCHENDU, A., J. CAO, Q. WANG, B. LUO, and D. LEE (2019) "Characterizing Man-made vs. Machine-made Chatbot Dialogs," in *Proceedings of the Int'l Conf. on Truth and Trust Online (TTO)*.

[121] SHAO, J., A. UCHENDU, and D. LEE (2019) "A reverse turing test for detecting machine-made texts," in *Proceedings of the 10th ACM Conference on Web Science*, pp. 275–279.

[122] OPENAI (2022) "Optimizing language models for dialogue," *https://openai.com/blog/chatgpt/*.

[123] BISWAS, S. (2023), "ChatGPT and the Future of Medical Writing," .

[124] DOWLING, M. and B. LUCEY (2023) "ChatGPT for (finance) research: The Bananarama conjecture," *Finance Research Letters*, p. 103662.

[125] COTTON, D. R., P. A. COTTON, and J. R. SHIPWAY (2023) "Chatting and Cheating. Ensuring academic integrity in the era of ChatGPT," .

[126] KIRCHENBAUER, J., J. GEIPING, Y. WEN, J. KATZ, I. MIERS, and T. GOLDSTEIN (2023) "A watermark for large language models," *arXiv preprint arXiv:2301.10226*.

[127] ABDELNABI, S. and M. FRITZ (2021) "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE, pp. 121–140.

[128] SHARMA, A., A. NANDAN, and R. RALHAN (2018) "An Investigation of Supervised Learning Methods for Authorship Attribution in Short Hinglish Texts using Char & Word N-grams," *arXiv preprint arXiv:1812.10281*.

[129] SARI, Y., A. VLACHOS, and M. STEVENSON (2017) "Continuous n-gram representations for authorship attribution," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 267–273.

[130] PROISL, T., S. EVERT, F. JANNIDIS, C. SCHÖCH, L. KONLE, and S. PIELSTRÖM (2018) "Delta vs. N-Gram Tracing: Evaluating the Robustness of Authorship Attribution Methods," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

[131] KESTEMONT, M. (2014) "Function words in authorship attribution. From black magic to theory?" in *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pp. 59–66.

[132] ZEČEVIĆ, A. (2011) "N-gram based text classification according to authorship," in *Proceedings of the Second Student Research Workshop associated with RANLP 2011*, pp. 145–149.

[133] LI, J., M. OTT, C. CARDIE, and E. HOVY (2014) "Towards a general rule for identifying deceptive opinion spam," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1566–1576.

[134] FERRACANE, E., S. WANG, and R. MOONEY (2017) "Leveraging discourse information effectively for authorship attribution," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 584–593.

[135] SUNDARARAJAN, K. and D. WOODARD (2018) "What represents "style" in authorship attribution?" in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2814–2822.

[136] HITSCHLER, J., E. VAN DEN BERG, and I. REHBEIN (2017) "Authorship attribution with convolutional neural networks and POS-Eliding," in *Proceedings of the Workshop on Stylistic Variation*, pp. 53–58.

[137] SEROUSSI, Y., I. ZUKERMAN, and F. BOHNERT (2014) "Authorship attribution with topic models," *Computational Linguistics*, **40**(2), pp. 269–310.

[138] SEROUSSI, Y., F. BOHNERT, and I. ZUKERMAN (2012) "Authorship attribution with author-aware topic models," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, Association for Computational Linguistics, pp. 264–269.

[139] SEROUSSI, Y., I. ZUKERMAN, and F. BOHNERT (2011) "Authorship attribution with latent Dirichlet allocation," in *Proceedings of the fifteenth conference on computational natural language learning*, Association for Computational Linguistics, pp. 181–189.

[140] HALVANI, O., L. GRANER, R. REGEV, and P. MARQUARDT (2020) "An Improved Topic Masking Technique for Authorship Analysis," *arXiv preprint arXiv:2005.06605*.

[141] UCHENDU, A., J. CAO, Q. WANG, B. LUO, and D. LEE (2019) "Characterizing Man-made vs. Machine-made Chatbot Dialogs." in *TTO*.

[142] ZHENG, R., J. LI, H. CHEN, and Z. HUANG (2006) "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**(3), pp. 378–393.

[143] VAN CRANENBURGH, A. (2012) "Literary authorship attribution with phrase-structure fragments," in *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pp. 59–63.

[144] HOENEN, A. and N. SCHENK (2018) "Knowing the Author by the Company His Words Keep," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

[145] SOLORIO, T., S. PILLAY, S. RAGHAVAN, and M. MONTES (2011) "Modality specific meta features for authorship attribution in web forum posts," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 156–164.

[146] TSCHUGGNALL, M. and G. SPECHT (2014) "Enhancing authorship attribution by utilizing syntax tree profiles," in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pp. 195–199.

[147] HOWEDI, F. and M. MOHD (2014) "Text classification for authorship attribution using Naive Bayes classifier with limited training data," *Computer Engineering and Intelligent Systems*, **5**(4), pp. 48–56.

[148] BARON, G. (2014) "Influence of data discretization on efficiency of Bayesian classifier for authorship attribution," *Procedia Computer Science*, **35**, pp. 1112–1121.

[149] HOU, R. and C.-R. HUANG (2017) "Stylometric Studies based on Tone and Word Length Motifs," in *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pp. 56–63.

[150] ALSHAHER, H. and J. XU (2020) "A New Term Weight Scheme and Ensemble Technique for Authorship Identification," in *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis*, pp. 123–130.

[151] BOUMBER, D., Y. ZHANG, and A. MUKHERJEE (2018) "Experiments with Convolutional Neural Networks for Multi-Label Authorship Attribution," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

[152] REN, Y. and D. JI (2017) "Neural networks for deceptive opinion spam detection: An empirical study," *Information Sciences*, **385**, pp. 213–224.

[153] ALSULAMI, B., E. DAUBER, R. HARANG, S. MANCORIDIS, and R. GREENSTADT (2017) "Source code authorship attribution using long short-term memory based networks," in *European Symposium on Research in Computer Security*, Springer, pp. 65–82.

[154] LÓPEZ-MONROY, A. P., F. A. GONZALEZ, and T. SOLORIO (2020) "Early author profiling on Twitter using profile features with multi-resolution," *Expert Systems with Applications*, **140**, p. 112909.

[155] SIMKO, L., L. ZETTLEMOYER, and T. KOHNO (2018) "Recognizing and imitating programmer style: Adversaries in program authorship attribution," *Proceedings on Privacy Enhancing Technologies*, **2018**(1), pp. 127–144.

[156] JUOLA, P. (2012) "Detecting stylistic deception," in *Proceedings of the Workshop on Computational Approaches to Deception Detection*, Association for Computational Linguistics, pp. 91–96.

[157] SÁNCHEZ-JUNQUERA, J., L. VILLASEÑOR-PINEDA, M. MONTES-Y GÓMEZ, P. ROSSO, and E. STAMATATOS (2020) "Masking domain-specific information for cross-domain deception detection," *Pattern Recognition Letters*, **135**, pp. 122–130.

[158] Manjavacas, E., J. De Gussem, W. Daelemans, and M. Kestemont (2017) “Assessing the stylistic properties of neurally generated text in authorship attribution,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pp. 116–125.

[159] Schürmann, T. and P. Grassberger (1996) “Entropy estimation of symbol sequences,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **6**(3), pp. 414–427.

[160] Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014) “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*.

[161] Zhang, X., J. Zhao, and Y. LeCun (2015) “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, pp. 649–657.

[162] Kim, Y. (2014) “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*.

[163] Koch, G., R. Zemel, and R. Salakhutdinov (2015) “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2, Lille.

[164] Sari, Y., M. Stevenson, and A. Vlachos (2018) “Topic or style? exploring the most useful features for authorship attribution,” in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 343–353.

[165] Stamatatos, E. (2017) “Authorship attribution using text distortion,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1138–1149.

[166] Pennington, J., R. Socher, and C. D. Manning (2014) “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

[167] Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang (2016) “SQuAD: 100,000+ Questions for Machine Comprehension of Text,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392.

[168] Wang, A., A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman (2018) “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,” in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355.

[169] Coyotl-Morales, R. M., L. Villaseñor-Pineda, M. Montes-y Gómez, and P. Rosso (2006) "Authorship attribution using word sequences," in *Iberoamerican Congress on Pattern Recognition*, Springer, pp. 844–853.

[170] Solaiman, I., M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, et al. (2019) "Release strategies and the social impacts of language models," *arXiv preprint arXiv:1908.09203*.

[171] Sapkota, U., S. Bethard, M. M. y Gomez, and T. Solorio (2015) "Not All Character N-grams Are Created Equal: A Study in Authorship Attribution," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ACL, Denver, Colorado, pp. 93–102.

[172] Fabien, M., E. Ú Villatoro-Tello, P. Motlicek, and S. Parida (2020) "BertAA: BERT fine-tuning for Authorship Attribution," in *Proceedings of the 17th International Conference on Natural Language Processing*, CONF, ACL.

[173] Goldstein, J., R. Winder, and R. Sabin (2009) "Person identification from text and speech genre samples," in *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 336–344.

[174] Uchendu, A., T. Le, and D. Lee (2023) "Attribution and Obfuscation of Neural Text Authorship: A Data Mining Perspective," *SIGKDD Explorations*, p. vol. 25.

[175] Port, A., I. Gheorghita, D. Guth, J. M. Clark, C. Liang, S. Dasu, and M. Marcolli (2018) "Persistent topology of syntax," *Mathematics in Computer Science*, **12**(1), pp. 33–50.

[176] Port, A., T. Karidi, and M. Marcolli (2019) "Topological analysis of syntactic structures," *arXiv preprint arXiv:1903.05181*.

[177] Munch, E. (2017) "A user's guide to topological data analysis," *Journal of Learning Analytics*, **4**(2), pp. 47–61.

[178] Turkes, R., G. F. Montufar, and N. Otter (2022) "On the Effectiveness of Persistent Homology," *Advances in Neural Information Processing Systems*, **35**, pp. 35432–35448.

[179] Perez, I. and R. Reinauer (2022) "The topological BERT: Transforming attention into topology for natural language processing," *arXiv preprint arXiv:2206.15195*.

[180] Reif, E., A. Yuan, M. Wattenberg, F. B. Viegas, A. Coenen, A. Pearce, and B. Kim (2019) "Visualizing and measuring the geometry of BERT," *Advances in Neural Information Processing Systems*, **32**.

[181] TENNEY, I., D. DAS, and E. PAVLICK (2019) "BERT Rediscovers the Classical NLP Pipeline," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601.

[182] ZHU, X. (2013) "Persistent homology: An introduction and a new text representation for natural language processing." in *IJCAI*, pp. 1953–1959.

[183] SAVLE, K., W. ZADROZNY, and M. LEE (2019) "Topological data analysis for discourse semantics?" in *Proceedings of the 13th International Conference on Computational Semantics-Student Papers*, pp. 34–43.

[184] DOSHI, P. and W. ZADROZNY (2018) "Movie genre detection using topological data analysis," in *Statistical Language and Speech Processing: 6th International Conference, SLSP 2018, Mons, Belgium, October 15–16, 2018, Proceedings 6*, Springer, pp. 117–128.

[185] HAGHIGHATKHAH, P., A. FOKKENS, P. SOMMERAUER, B. SPECKMANN, and K. VERBEEK (2022) "Story Trees: Representing Documents using Topological Persistence," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 2413–2429.

[186] WU, X., X. NIU, and R. RAHMAN (2022) "Topological Analysis of Contradictions in Text," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2478–2483.

[187] CHERNIAVSKII, D., E. TULCHINSKII, V. MIKHAILOV, I. PROSKURINA, L. KUSHNAREVA, E. ARTEMOVA, S. BARANNIKOV, I. PIONTKOVSKAYA, D. PIONTKOVSKI, and E. BURNAEV (2022) "Acceptability judgements via examining the topology of attention maps," *arXiv preprint arXiv:2205.09630*.

[188] TYMOCHKO, S., Z. NEW, L. BYNUM, E. PURVINE, T. DOSTER, J. CHAPUT, and T. EMERSON (2020) "Argumentative topology: Finding loop (holes) in logic," *arXiv preprint arXiv:2011.08952*.

[189] MCGUIRE, S., S. JACKSON, T. EMERSON, and H. KVINGE (2023) "Do neural networks trained with topological features learn different internal representations?" in *NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, PMLR, pp. 122–136.

[190] AI, B., Y. WANG, Y. TAN, and S. TAN (2022) "Whodunit? Learning to Contrast for Authorship Attribution," in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pp. 1142–1157.

[191] CLARK, E., T. AUGUST, S. SERRANO, N. HADUONG, S. GURURANGAN, and N. A. SMITH (2021) "All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Online, pp. 7282–7296.
URL `https://aclanthology.org/2021.acl-long.565`

[192] DOU, Y., M. FORBES, R. KONCEL-KEDZIORSKI, N. A. SMITH, and Y. CHOI (2022) "Is GPT-3 Text Indistinguishable from Human Text? Scarecrow: A Framework for Scrutinizing Machine Text," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7250–7274.

[193] MARTIN, F., T. SUN, M. TURK, and A. D. RITZHAUPT (2021) "A meta-analysis on the effects of synchronous online learning on cognitive and affective educational outcomes," *International Review of Research in Open and Distributed Learning*, **22**(3), pp. 205–242.

[194] HRASTINSKI, S. (2008) "The potential of synchronous communication to enhance participation in online discussions: A case study of two e-learning courses," *Information & Management*, **45**(7), pp. 499–506.

[195] PETERSON, A. T., P. N. BEYMER, and R. T. PUTNAM (2018) "Synchronous and asynchronous discussions: Effects on cooperation, belonging, and affect." *Online Learning*, **22**(4), pp. 7–25.

[196] IPPOLITO, D., D. DUCKWORTH, C. CALLISON-BURCH, and D. ECK (2020) "Automatic Detection of Generated Text is Easiest when Humans are Fooled," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, pp. 1808–1822.
URL `https://aclanthology.org/2020.acl-main.164`

[197] VAN DER LEE, C., A. GATT, E. VAN MILTENBURG, S. WUBBEN, and E. KRAHMER (2019) "Best practices for the human evaluation of automatically generated text," in *Proceedings of the 12th International Conference on Natural Language Generation*, pp. 355–368.

[198] FLEISS, J. L. (1971) "Measuring nominal scale agreement among many raters." *Psychological bulletin*, **76**(5), p. 378.

[199] MERCIER, H. and D. SPERBER (2011) "Why do humans reason? Arguments for an argumentative theory." *Behavioral and brain sciences*, **34**(2), pp. 57–74.

[200] BIRNHOLTZ, J. and S. IBARA (2012) "Tracking changes in collaborative writing: edits, visibility and group maintenance," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 809–818.

[201] SHIRANI, A. I., M. H. TAFTI, and J. F. AFFISCO (1999) "Task and technology fit: a comparison of two technologies for synchronous and asynchronous group communication," *Information & management*, **36**(3), pp. 139–150.

[202] MABRITO, M. (2006) "A study of synchronous versus asynchronous collaboration in an online business writing class," *The American Journal of Distance Education*, **20**(2), pp. 93–107.

Vita

Adaku Uchendu

Education

Pennsylvania State University *State College, Pennsylvania*
Ph.D. in Information Sciences and Technology *August 2018 - August 2023*
University of Maryland Baltimore County *Baltimore, Maryland*
B.S. Mathematics | Minor: Statistics, **Honors**: Cum Laude *August 2014 - May 2018*

Publications

1. **Adaku Uchendu**, Thai Le, Dongwon Lee, "TopRoBERTa: Topology-Aware Authorship Attribution of Neural Texts," *Under Review*, 2023
2. **Adaku Uchendu***, Jooyoung Lee*, Hua Shen, Thai Le, Kenneth Huang, Dongwon Lee, "Understanding Individual and Team-based Human Factors in Detecting Deepfake Texts," *arXiv preprint arXiv:2304.01002.*, 2023
3. **Adaku Uchendu**, Thai Le, Dongwon Lee, "Attribution and Obfuscation of Neural Text Authorship: A Data Mining Perspective," *SIGKDD Explorations*, Vol. 25, June 2023
4. **Adaku Uchendu**, Daniel Campoy, Christopher Menart, Alexandra Hildenbrandt, "Robustness of Bayesian Neural Networks to White-Box Adversarial Attacks," *IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering*. IEEE, 2021.
5. **Adaku Uchendu**, Zeyu Ma, Thai Le, Rui Zhang, Dongwon Lee. "TuringBench: A Benchmark Environment for Turing Test in the Age of Neural Text Generation," *Findings of the Association for Computational Linguistics: EMNLP 2021*, November 2021
6. **Adaku Uchendu**, Thai Le, Kai Shu, Dongwon Lee. "Authorship Attribution for Neural Text Generation," *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Virtual Event, November 2020
7. **Adaku Uchendu**, Jeffrey Cao, Qiaozhi Wang, Bo Luo, Dongwon Lee. "Characterizing Man-made vs. Machine-made Chatbot Dialogs," *Truth and Trust Online (TTO)*, London, UK, October 2019