Threat Scenarios and Best Practices for Neural Fake News Detection

Artidoro Pagnoni[†] and **Martin Garciarena**[◊] and **Yulia Tsvetkov**[†]
[†]Paul G. Allen School of CSE, University of Washington [◊] SRI International
{artidoro, yuliats}@cs.washington.edu, martin.graciarena@sri.com

Abstract

In this work, we discuss different threat scenarios from neural fake news generated by state-of-the-art language models. Through our experiments, we assess the performance of generated text detection systems under these threat scenarios. For each scenario, we also identify the *minimax* strategy for the detector that minimizes its worst case performance. This constitutes a set of best practices that practitioners can rely on. In our analysis, we find that detectors are prone to shortcut learning (i.e., lack of out-of-distribution generalization) and discuss approaches to mitigate this problem and improve detectors more broadly. Finally, we argue that strong detectors should be released along with new generators.[1]

1 Introduction

During the COVID-19 pandemic, an overabundance of inaccurate information made it hard for people to find reliable guidance when they needed it, creating the first global *infodemic*. This widespread presence of online fake news poses a risk for the principles of the global information and communication space, which is considered to be a public good (Stiglitz et al., 1999). Widely available and reliable information is necessary for the function of a democratic state and is seen by UNESCO as vital to ensure "public participation and civic space".[2]

Neural generation systems unlock new possibilities for disinformation campaigns to generate large quantities of targeted content (Zellers et al., 2019; Radford et al., 2019; Keskar et al., 2019; Brown et al., 2020). Recent progress in pretrained language models enables the generation of fluent, original text that can be easily confused with human-written text. Progressively more complex use of such language models can be achieved depending on the technical expertise and resources of the users (i.e., access to the language model, capability to run existing systems, to finetune them, or to develop a new generator and train it). This is a pressing problem, and it is reported that certain actors spend substantial monetary resources to spread misinformation and influence public opinion, for example during the COVID 19 pandemic [3] and the US Presidential Elections of 2016 (Badawy et al., 2018; Stiff and Johansson, 2021). Although the majority of online disinformation is currently manually written (Vargo et al., 2018), the rapid progress of language generation systems along with their increased availability could attract these actors for whom the financial barriers do not pose problems. To prevent the spread of artificial and potentially malicious content online, it is necessary to develop automated detection systems that can distinguish human and generated text reliably.

In this work, we discuss the problem of machine generated text detection (Jawahar et al., 2020), by identifying and analyzing potential scenarios of malicious text generation (i.e., threat scenarios) based on (1) state-of-the-art models in neural language generation, and (2) previous work that describes how these techniques can be used to forge content (Ranade et al., 2021; Gupta et al., 2020; Buchanan et al., 2021).

Our analysis focuses on the interaction of generators and detectors for each scenario across their technical capabilities and associated costs. Knowing the limitations of current detection systems can help policymakers identify the threats posed by different actors by assessing their technical expertise and resources.

The contributions of this work are: (1) identifying and assessing threat scenarios for the use of pretrained language models to spread disinforma-

[1]Resources including generators, detectors, and generated and pristine news articles are made available at `https://github.com/artidoro/detect-gentext`

[2]UNESCO report "Information as a public good"

[3]New York Times: "Disinformation for Hire, a Shadow Industry, Is Quietly Booming"

tion, based on associated costs and availability of generators, (2) analyzing the performance of current methods – discriminators trained to defend against these threat scenarios, and (3) identifying challenges and future research directions to mitigate potential threats.

2 Problem Description

In this section we introduce the problem of detecting generated text and describe state-of-the-art methods for both generators and detectors.

2.1 Generators

We consider a generator to be a model capable of generating fluent text. Currently, state-of-the-art generators rely on neural language models pretrained on large text corpora (Bengio et al., 2000). Language models generate text autoregressively, by sampling each word from a probability distribution over a fixed vocabulary given the word's context.

Architecture and Training Objective The core of the architecture of neural language models has converged to Transformer models (Vaswani et al., 2017) with small variations (Press et al., 2022). One distinction between language models is their training objective (ex: *masked* (Devlin et al., 2018) and *causal* (Radford et al., 2019) language models). Causal language models are generally the ones employed for text generation (Li et al., 2021).

Training Data Language models are trained to mimic the distribution of the language in their training data. This has an impact on the kind of text they are able to generate. Language models pick up social biases and other unwanted artifacts of the text they were trained on (Sun et al., 2019; Weidinger et al., 2021; Field et al., 2021). Most language models are pretrained on large corpora of web text. They can be adapted to specific domains by finetuning on new corpora. For example, a model could be finetuned to the news domain to generate text in the style of news articles (Zellers et al., 2019).

Sampling From the Language Model There are different strategies to sample from the distribution produced by the language model. These strategies lead to different styles of text. *Greedy sampling* amounts to generating the highest probability word. This strategy, however, leads to deterministic generations. *Random sampling* involves sampling from the entire distribution over the vocabulary. In some cases, this can lead to ungrammatical generations. To mitigate this issue *top-k sampling* (Fan et al., 2018) generates from the top-k most likely words according to the language model. Finally, *nucleus sampling* (Holtzman et al., 2019) samples from the set of words that collectively accounts for a probability mass p under the language model. Orthogonal to sampling strategies are decoding algorithms that aim to maximize the global probability over the full text sequence. Most generators compose text sequences one word at a time without guarantees of finding a globally optimal sequence. Beam search is a heuristic method that is often employed to maintain b candidate generations and pick the one with the highest overall likelihood.

2.2 Detectors

In this work, we define detectors as models that are trained to distinguish machine-generated and human-written text. Research on detectors for generated text is motivated by two main goals. On one hand, a better understanding of the differences between generated and human text can be used to improve generated systems. On the other hand, better detection systems could help mitigate the potential negative societal impact that machine generated text can have (Jawahar et al., 2020).

Previous work has pointed out stylistic differences between generated and human text. Differences in fluency, lexical, and syntactic novelty have all been shown to help differentiate between generated and human written text. Here, we describe different lines of work on automated detection systems of machine generated text with a particular focus on pretrained classifiers which have been shown to work best so far (Gehrmann et al., 2019; Dugan et al., 2020).

Human-machine collaboration Dugan et al. (2020) show that humans can easily be fooled when trying to detect the boundary between human and generated text. Their work highlights the difficulty of the detection task, but also shows that humans are capable of identifying generated text in a long document (beyond a couple of sentences). Gehrmann et al. (2019) studied the difference between human and machine generated text. They showed that generated text tends to opt for more common phrases, thus favoring fluency over novelty. They built a tool, GLTR, that helps humans identify generated text, by visualizing the likelihood and rank of words in a text under a language model.

Description	Costs	Expertise	Threat	Detect. Acc.
Online playgrounds	0	Interacting with an online form	Execute Generator	92.8%
Paid API	0-1K	Prompting and querying API	Execute Generator	92.8%
Execution of LM	50k-130k	Hardware setup, DL execution experience	Execute Generator	92.8%
Finetuning of LM	100k+	Hardware setup, DL training expertise	Finetune Generator	74.2%
Training of LM	100k-1M+	Hardware setup, optimization, training of LM	Train Generator	64.8%

Table 1: Threat scenarios, costs, and *minimax* detector performance (see discussion in section 6).

Stylometry Aiming to identify differences in the style of generated and human text, some approaches directly measure stylistic features of text. These approaches have broadly been termed stylometry. Some features used by Fröhling and Zubiaga (2021) include repetitiveness, lack of purpose, and readability. Despite some success, previous work pointed out the limitations of stylometry in detecting machine-generated fake news (Schuster et al., 2020). State-of-the-art performance on this task is achieved by pretrained language models finetuned to the classification task (Uchendu et al., 2021).

Pretrained classifiers Previous work has studied how well pretrained classifiers can detect generated text (Solaiman et al., 2019; Zellers et al., 2019; Ippolito et al., 2020; Bakhtin et al., 2021; Uchendu et al., 2021). Solaiman et al. (2019) show that finetuning RoBERTa can detect GPT-2 (Radford et al., 2019) with 95% accuracy and that the performance transfers across decoding strategies and to smaller generators. Ippolito et al. (2020) show that detectors perform best when humans are fooled because decoding strategies have to compromise between fluency and lexical and syntactic novelty. Fluency errors are easy to detect by humans, while lexical novelty is difficult. On the other hand, lexical novelty is what classifiers are able to identify.

3 Threat Scenarios

A given generator can be used in various ways, based on domain expertise and budget, resulting in different styles of generated text. We describe threat scenarios by identifying possible uses of pretrained LMs and their associated costs (technical expertise and monetary resources required).

Interactive Playgrounds and Paid APIs Most language models have been open-sourced and the model parameters can be downloaded. However, many also have free online playgrounds that simplify the user-interaction[4] and require little to no programming expertise. The latest models coming from the private sector (e.g., GPT3 (Brown et al., 2020)) are usually not available publicly or are only accessible through paid APIs and online playgrounds. With some delay, equally powerful models are often released publicly (e.g., OPT (Zhang et al., 2022)).

These playgrounds are either free or very cheap. Even the paid GPT3 playground's most powerful model only costs $0.06 for 1000 tokens. Generating text through the playgrounds is time consuming and requires heavy intervention by the user who needs to write prompts and ensure that the generation is sensible. However, this method could be used to speed up the process of writing fake content by offloading some portion of the writing to the language model. APIs extend the playgrounds by allowing generation to be performed programmatically. This increases the potential scale of the generation process. Targeted generations through prompting still need careful handcrafting.

Execution and Sampling Variations The next barrier to using pretrained language models is being able to execute them. Due to their scale language models require hardware accelerators (ex: GPUs and TPUs). Thanks to advances in model parallelization (Rasley et al., 2020; Rajbhandari et al., 2020) it is no longer required to store the full model in the memory of a single piece of hardware. This makes it possible to execute large models on standard GPU hardware.

Once the model is executing, it is simple to vary hyperparameters of the generation process such as sampling strategy, temperature, beam search, and repetition penalty. These parameters are generally built into the software libraries for text generation and lead to stylistic differences in the generated text which can reduce the effectiveness of detection systems. Having the monetary and technical resources to execute the models allows generat-

[4]https://transformer.huggingface.co/

ing text at scale. However, targeted generations still require prompts either curated or automatically extracted. Previous work puts the cost of such a project including experimentation and servicing between \$50,000 and \$136,000 (Sharir et al., 2020). These costs will likely decrease as hardware becomes more easily accessible and software packages and cloud services simplify the process.

Finetuning To produce more sophisticated and targeted generations, it is usually necessary to adapt the pretrained language model. One common technique for adaptation of generators is finetuning the generators by continuing the pretraining (Ranade et al., 2021; Gupta et al., 2020). The result is a language model that is able to produce text in the required style, format, or domain. This requires both access to in-domain data as well as computing resources. Resources required for this are upwards of \$100,000 (Sharir et al., 2020).

Training a Language Model Developing new models can lead to new and unique styles of generation. These can be hard to identify for detection systems that have not seen such examples in their training data. Training new language generation systems comes at a significant cost. The costs are inflated by the need of several training runs to experiment with different hyperparameter configurations. Previous research estimated the costs of training new models at \$50k for a 110 million parameter model, \$200k for a 340 million parameter model, and \$1.6M for a 1.5 billion parameter model (Sharir et al., 2020). Note that these costs were estimated in 2020 and are likely going to be affected by improvements in hardware and software libraries to speed up the training process. Costs are even higher for larger models like OPT (175 billion param.) which was trained on 992 80GB A100 GPUs with experiments lasting at least two months (Zhang et al., 2022). At the time of writing, the GPU hardware alone would cost \$50-100M on the public version of Google cloud (although due to private negotiations with the cloud providers the actual price is likely lower).

4 Experimental Setup

Here we describe the generators, detectors, and datasets that we use in our experiments.

Model Name	Num Param.	LAM PPL	LAM Acc	Detect. Acc.
GPT-2 sm	125M	35.1	45.99	99.49
GPT-2 md	350M	15.6	55.48	98.79
GPT-2 lg	760M	10.9	60.12	97.39
GPT-2 xl	1.5B	8.6	63.24	95.91
GPT-NEO	2.7B	5.6	62.2	95.12
GPT-3	175B	3.0	76.2	92.83

Table 2: Generators used in our experiments, the number of parameters, their perplexity, and accuracy on the LAMBADA dataset (Paperno et al., 2016), and their in-domain detection performance when training and testing with nucleus sampling ($p = 0.96$).

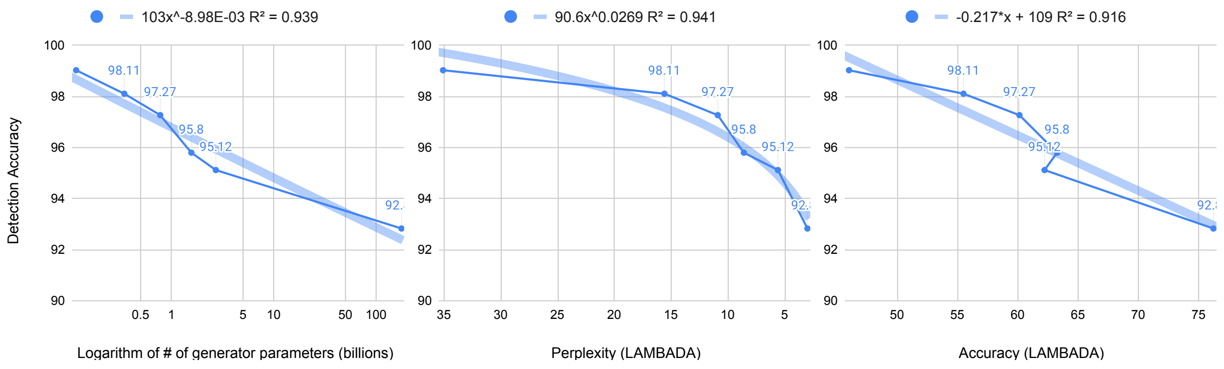
4.1 Generators

In this work, we focus on causal language model generators. These have been shown to be successfully employed to generate neural fake news (Zellers et al., 2019). We experiment with the four sizes of GPT-2 (Radford et al., 2019), GPT-NEO (Black et al., 2021), and GPT-3 (Brown et al., 2020). We chose these models as they have a similar training objective and cover a wide range of performances in terms of perplexity. The details of these models are shown in Table 2.

Our experiments do not include larger models (Rae et al., 2021; Smith et al., 2022; Chowdhery et al., 2022). The perplexity and accuracy on benchmarks like LAMBADA (Paperno et al., 2016) of the larger model are not significantly different. For example, the accuracy of PALM (540B parameters) on LAMBADA is 77.9 against 76.2 for GPT-3 (175B parameters). We also do not consider controllable text generation systems which can change the domain of the generated text. Finetuning is a simple alternative to change the domain and we chose it to avoid introducing other confounders. Also, previous work that experiment with controllable generation did not always find significant differences in detection performance (Stiff and Johansson, 2021).

4.2 Detectors

In this work, we experiment with classifiers based on pretrained language models and finetuned to the task of detecting generated text. In total, we experiment with seven different detectors based on ROBERTA(Liu et al., 2019), BERT(Devlin et al., 2019), ALBERT (Lan et al., 2020), and ELECTRA (Clark et al., 2020). We include different



(a) (b) (c)

Figure 1: In-domain detection performance as a function of number of parameters (a), perplexity (b), and accuracy (c) of the generator (perplexity and accuracy on LAMBADA dataset). Detectors are trained and tested on nucleus sampling ($p = 0.96$).

sizes and training objectives for the pretrained language models that the detectors rely on.

4.3 Datasets

We employ existing dataset of generated text for GPT-2 models and complement it with generations from GPT-NEO and GPT-3, as well as domain adapted generations. OpenAI released both human and machine generated text.[5] The data includes generations from GPT-2 models along with part of the original training data (Solaiman et al., 2019). The generations come with three different decoding strategies (random, top-k, and nucleus sampling). We are not aware of any other models being released with sample generations. Furthermore, the pretraining data is often not available. We therefore had to generate a large training dataset of text generated by GPT-NEO and GPT-3 for our experiments. We use human prompts (15 tokens) from Webtext (released by OpenAI) and the three sampling strategies that were already used by OpenAI. The hyperparameters $k = 40$ and $p = 0.96$ were reported to be a good compromise between fluency and novelty of generations (Holtzman et al., 2019) and are used in our experiments. For GPT-3, we collected 50K training examples of generated text using nucleus sampling with $p = 0.96$. To our knowledge, this is the first collection of documents that is large enough to train a detector for GPT-3. To test generalization performance of detectors, all generators besides GPT-3 were also finetuned to the COVID domain. The dataset used for finetuning is the NELA 2020 (Gruppi et al., 2022) news corpus (the COVID split). Details of the datasets used in our experiments are available in the appendix Appendix A.

[5]https://github.com/openai/gpt-2-output-dataset

5 Results

Our experiments simulate the different threat scenarios identified in the previous section and evaluate their associated risks. Borrowing from the field of security (Shostack, 2014), we describe how "attackers" can employ generators and "defenders" train effective detectors under varying assumptions.

The first three scenarios (online playgrounds, paid APIs, and execution of LM) can be studied together as they involve using an existing generator without any domain specific tuning of the model parameters. Users can still tailor the generation style by changing decoding hyperparameters. Next, we explore the scenario of generator finetuning, focusing on the covid domain. Lastly, we simulate the training of new generators by looking at the performance of detectors when they are trained and tested on different generators. Throughout our experiments, we choose to isolate and study a specific phenomenon to determine its impact of the detection performance in isolation.

5.1 Preliminary

To set the basis for further experiments, we begin with a scenario where there is full knowledge of the attack. Here, both training and testing is performed on the same generator and decoding strategy while finetuning. This is a white-box attack which, though unrealistic, gives some useful preliminary information about the hardness of the task.

tokens	256	128	64	32
GPT-2 xl nucleus	96.4	93.09	85.7	76.37
GPT-2 xl random	95.96	94.4	87.9	76.69
GPT-2 xl $k = 40$	97.9	98.8	96.8	-

Table 3: Accuracy on shorter sequences. RoBERTa large was trained and tested on sequences of varying size in terms of tokens.

train $\downarrow$ test $\rightarrow$	random	top-k	nucleus
random	n.a.	-28.6	-21.9
top-$k = 40$	-43.2	n.a.	-21.3
nucleus $p = 0.96$	-8.6	$+2.4$	n.a.

Table 4: Average accuracy variation when training a RoBERTa lg model on one decoding strategy and testing on another decoding strategy across the generators of interest.

Under these circumstances, the detection performance is above 90% even for the largest generator GPT-3 (Table 2). We confirm observations made by previous work (Radford et al., 2019; Ippolito et al., 2020) that larger detectors perform better (-4.6% average performance decrease with the small version of the detector across 60 generator variations tested) and that larger models are harder to detect. A contribution of our analysis is a study of the scaling laws of generator capacity and detection performance. We can perform such a study thanks to the collection of a dataset of GPT-3 generations.

In Figure 1, we observe that the detection performance is related with a power law to both the number of parameters in the generator and the generator perplexity with an exponent in the order of 10^{-3}. The relation appears to be more linear in terms of the accuracy on the LAMBADA dataset with coefficient -0.21. In all three cases, we have $R^2 > 0.9$. These relations suggest a trend with small detection performance drop as the generator complexity increases even beyond the current known generators. As a reminder, this assumes a white-box attack scenario.

Besides having full knowledge of the attack, the previous results also assumed fairly long generations. The detection performance, even under white-box attack, significantly decreases when the text sequences to distinguish are shorter. Such shorter text sequences are frequent in certain domains like in social media (Twitter has 280 character limit which is about 50 words). In Table 3, we show that on sequences of 64 tokens, roughly the maximum length of a Tweet, performance drops by 10% on nucleus and random sampling. Interestingly, on top-k the decrease in performance is not as pronounced.

5.2 Executing Known Generators

In this scenario, we assume that attackers can only employ generators that are known to the defenders, which means that the adversary can pick a decoding strategy. Since the generator is known, the defender can train the detector on data generated by the generator and used by the adversary. However, given the large number of available combinations of decoding hyperparameters, the defender cannot assume to have trained a model on the same decoding strategy that was chosen by the adversary.

In practice, we simulate this scenario by training the detector on one decoding strategy and testing it on another. In Table 4, we show that performance generally decreases when a detector is evaluated on a decoding strategy it was not trained on. This highlights how sensitive these models are to small stylistic variations of the generated text. This experiment also shows that nucleus sampling has the best generalization to other decoding strategies and that defenders should train on nucleus sampling to mitigate worst case performance drop in this scenario. Training on nucleus sampling still has an average performance change of -8.6% when testing on random sampling. Such significant reduction indicates that changes in the decoding strategy are simple ways to reduce the performance of a detector.

5.3 Finetuning Generators

To simulate the scenario of a finetuned generator, we assume that the adversary can only finetune a known generator. In our case, we finetune the generators to the COVID news domain using masked language modeling. Here, the defender does not know the target domain, therefore the detectors can only be trained in the original domain of the generator (generic web text).

In Table 5, we compare in-domain (ID) performance, where RoBERTa large is trained and tested on data from the same generator, with out-of-domain (OOD) performance, where RoBERTa lg and ELECTRA lg were trained on GPT-2 xl outputs in the web domain and tested on generated text in the COVID news domain. Here we keep the sampling strategy fixed (nucleus sampling $p = 96$)

Generators (COVID)	ID Acc.	OOD Acc.	
		RoBERTa lg	ELECTRA lg
GPT-2 sm	99.72	75.86	95.24
GPT-2 md	99.87	75.86	89.14
GPT-2 lg	98.59	90.65	94.37
GPT-2 xl	95.47	69.59	78.56
GPT-Neo	96.16	79.46	74.27
Average	97.96	81.45	86.31

Table 5: Detection accuracy in the COVID domain. We compare in-domain (ID) with out-of-domain (OOD) performance for different detectors.

to isolate the change in performance due to a shift to the COVID news domain.

We find that the out-of-domain performance is on average 11% lower for ELECTRA lg and 16% lower for RoBERTa lg compared to the in-domain performance. The ELECTRA discriminator is pretrained to identify spans of text that were replaced by a language model. This pretraining objective is closer to the task of detecting generated text than the masked language modeling objective used by RoBERTa. We hypothesize that ELECTRA performs better at generalizing to out-of-domain generated text because of its training objective.

5.4 Training New Generators

To test the performance under new and unknown generators, we make the assumption that the defender does not have access to the generator it is trying to detect. This means that the defender cannot train on the target generator.

To simulare this scenario, we test the detectors on generators they were not trained on. To isolate this component (i.e., change in generator), we do not vary the decoding strategy or the domain of the generator. In Table 6, we show that there is a drop in performance across all detectors when they are trained on a smaller generator and tested on a larger one. This drop is more pronounced when the detectors are trained on smaller models (decrease of 26.5% when a model trained on the GPT-2 small is tested on GPT-2 xl). However, when comparing very large models the difference in performance is significantly smaller (training on GPT-2 lg and testing on GPT-xl leads to only a 2.1% performance drop). We conclude that the generalization benefit from using a larger generator for training decreases once the generator is above a certain quality. This observation is in line with our study of the scaling law of the capacity of the generator under the white-box attack.

Trained on →	sm			md		lg
Tested on →	xl	lg	md	xl	lg	xl
RoBERTa large	36.8	22.3	6.9	11.8	2.8	2.0
RoBERTa base	35.5	24.6	13.3	11.3	4.0	1.9
BERT large	31.2	22.7	16.3	10.7	4.4	2.2
BERT base	16.5	11.4	12.0	6.5	3.3	1.3
ELECTRA small	15.5	11.2	9.0	8.8	5.1	1.1
ALBERT base	23.4	17.2	12.0	12.5	6.9	4.2
Avg. acc. change	26.5	18.2	11.6	10.3	4.4	2.1

Table 6: Decrease in accuracy when training on smaller generators and testing on larger ones. The change is calculated based on the performance of the discriminator tested on the same generator size it was trained on. Here we use different sizes of GPT-2 with nucleus sampling.

	GPT-2 lg	GPT-2 md	GPT-2 sm
RoBERTa lg	1.0	1.2	1.4
RoBERTa base	1.2	1.4	2.2
BERT lg	1.0	0.3	1.8
BERT base	0.8	-2.9	0.0
ELECTRA sm	0.5	-3.3	-0.4
ALBERT	1.7	-0.1	4.4

Table 7: Training on larger generators transfers to smaller ones. In this case, the discriminators trained on GPT-2 xl (nucleus sampling) maintain accuracy on the smaller generators.

Next, we test the hypothesis that training on larger generators transfers to smaller generators. In Table 7, we show the performance change when training on GPT-2 xl and testing on GPT-2 lg, md, and sm. We observe that the performance, in terms of accuracy, remains mostly unchanged across detectors with a 0.2% average improvement. This indicates that training on larger generators generalizes to the smaller ones (see Appendix A for a discussion of the GPT-3 exception).

6 Discussion

6.1 Best Practices to Detect Generated Text

For each threat scenario we identify the *minimax* strategy (Nash, 1953) that minimizes the worst case performance for the defender (see Table 1). In general, our preliminary experiments show that it is beneficial to use the best available generator for training and the largest detection model. We therefore recommend this to practitioners and assume it is done by the defender in the following scenarios.

Executing Known Generators Online playgrounds, paid APIs, and models available for download and execution offer roughly the same language models (GPT-3 was not available for execution but OPT (Zhang et al., 2022) recently filled the gap). Our experiments as well as previous work show that nucleus sampling generalizes to other sampling techniques. When the generator is known, detectors should be trained on this sampling strategy. The detection performance of GPT-3 was the lowest at 92%.

Finetuning Known Generators When confronted with generators finetuned to new domains, we showed that ELECTRA generally performed better than other classifiers (possibly due to its training objective) and that it should be preferred over RoBERTa. The worst case detection performance for the ELECTRA large model was 74.2% with the finetuned GPT-Neo generator.

Training New Generators Finally, here we assume that the adversary has the resources required to train a new generator altogether. The worst case detection is achieved on the GPT-3 generator so we take that as the new generator. The RoBERTa large model trained on the otherwise largest available generator has a detection accuracy of 64.8%.

6.2 Detection Challenges

We found that detectors for generated text are highly prone to shortcut learning. This phenomenon seems to arise in many learning systems (Geirhos et al., 2020) and involves taking shortcuts that achieve high performance on a given benchmark or domain but fail to generalize and transfer to real-world scenarios.

Since the training domain of the generators is not always available, there can be a mismatch between the domain of the generated text and the human text that are used in the training dataset for detectors. Such small systematic differences between the generated text and the human written text can be picked up by the detectors and prevent them from generalizing to other generators or domains.

In our experiments, we observed that detection models seem to focus on stylistic confounds between the human and generated text when they are present. We observed this phenomenon with small differences in tokenization and domain. For example, the NELA dataset is provided pre-tokenized which leads to punctuation having additional white spaces compared to the generator outputs. A detector trained on this dataset achieves misleading near perfect performance (99.9% accuracy) but generalized poorly to datasets that did not have this tokenization mismatch (60% accuracy). We used SHAP (Lundberg and Lee, 2017) to find what the detector focused on and find that the human text had white spaces before punctuation due to the way it was collected and anonymized (see Figure 2). This is related to observations made in previous work which highlighted the vulnerability of detection systems to adversarial examples where simple changes in the text (e.g., changing characters) can flip the predictions of the models (Darmetko; Jun et al.; Wolff and Wolff, 2020; Gagiano et al., 2021).

Model	Acc.
Baseline (no ensemb.)	94.4
Majority Voting	95.5
Decision Tree	95.7
Logistic Reg.	**96.0**

Table 8: Ensembling specialized detectors.

6.3 Future Work

Interpretability Future work should investigate whether SHAP or other interpretability methods can be used to understand what the detectors rely on to discriminate generated and human-written text. Any findings could help further improve detectors and would also be valuable to the natural language generation community.

Ensembling We also experiment with ensembling specialized detectors instead of using a single detector. In Table 8, we show that ensembling detectors trained on different domains can lead to a more general detection system. We combine detectors trained on outputs of GPT-2 xl with three different decoding strategies. We observe that ensembling consistently leads to performance improvements over the baseline which uses a single detector trained on nucleus sampling (showed to generalize well to other sampling strategies). In this experiment, we test on data generated with all three decoding strategies.

We believe that combining specialized detectors is a promising area for future work. Interesting approaches might involved identifying the right detector to use for a given text which is related to the generator attribution task (Uchendu et al., 2020). They could also try to combine specialized detectors using model distillation (Hinton et al., 2015) into a single detector that learns from each specialized model on different training instances depending on which generator originated them.

Better Training Data We found that generators were sensitive to generation hyperparameters. Generating high quality training datasets required some generator-specific hyperparameter tuning. The absence of automated evaluation metrics of generated text makes the process laborious and time-consuming. Future work should draw from stylometry (Fröhling and Zubiaga, 2021) or unsupervised distributional methods (Pillutla et al., 2021; Gallé et al., 2021) to help automate the process of constructing a high quality training dataset. Features used for stylometry could more generally be used to identify a domain mismatch between human and generated text which if addressed could help improve the generalization performance of detectors.

Adversarial Setup Detecting generated text involves an adversarial setup in which detectors need generated outputs during training. The best training signal for a generator comes from the best generator. However, improvements in generators leave the detectors behind. One interesting avenue for future work is to adversarially filter the training data for the detector by only selecting examples that are most similar to human text (Holtzman et al., 2018). The process is similar to simulating a better generator with a simpler one. The filtering could be repeated iteratively and rely on a detector that is progressively trained on harder examples.

7 Ethical Considerations, Limitations, and Recommendations

Although in this work we discuss the problem of detecting generated text with a particular focus on generated fake news, we would like to point out that not all generated text is fake news and most generated text does not have a malicious intent. Language models are widely used in NLP for both conditional language generation tasks like summarization, translation, and dialogue (Gatt and Krahmer, 2018), as well as unconditional generation tasks like story generation (Fan et al., 2018).

In this work, by presenting results on the detection performance of different generators, we inevitably provide information that could inform malicious actors about how to improve generators and where our detection system fail. However, we believe that the experimental evidence in this paper points to research directions which can help improve systems for the detection of generated text beyond our current capabilities.

We also would like to point out that the findings of this work are limited to the domains we focused on but the methodology is general and can be applied to other domains. In particular, the out-of-domain performance will change depending on the distance between domains. For example, we did not include any experiments with other dialects of English or other languages.

One limitation of this work is that we do not experiment with controllable generation systems (Keskar et al., 2019; Krause et al., 2021; Dathathri et al., 2020; Kumar et al., 2021). This is partly due to the high risk of dual use in developing systems that controllably generate malicious content. We point to prior work which experimented detecting generations from these systems (Stiff and Johansson, 2021).

Finally, based on the observations made in this paper, we recommend that **language models should be released along with strong detectors**. Our experiments showed that in-domain detection performance is reasonably high ($>$ 90% acc.) even for the largest generator GPT-3, while the out-of-domain performance is significantly lower (64.8% acc.). We also pointed out that variations in the domain between generated data and human data used to train discriminators can lead to shortcut learning. This suggests that **the requirement for authors of new generators should be to release data sampled from the training dataset along with sample generations** to ensure the two splits of the data have matching domains.

8 Conclusion

We provide an assessment of the current landscape of generated text detection and identify three primary threat scenarios. Through extensive experimentation simulating the identified threats, we establish the *minimax* strategies that minimize the worst case scenario for the detector. We argue that these strategies constitute best practices for practitioners. We find that, when confronted with an adversary capable of training a new generator, the worst case detection performance could be as low as 64.8%. We then discuss observed detection challenges related to shortcut learning, point to several avenues for future work, and provide recommendations for the community to release detectors along with new models.

Acknowledgements

The authors are grateful to the anonymous reviewers for their comments, and to Evangelia Spiliopoulou, Gian Marco Visani, and the members of the Tsvetshop group for their invaluable feedback. This material is based upon work supported by the DARPA CMO under Contract No. HR001120C0124. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily state or reflect those of the United States Government or any agency thereof.

References

Adam Badawy, Emilio Ferrara, and Kristina Lerman. 2018. Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign. In *2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM)*, pages 258–265. IEEE.

Anton Bakhtin, Yuntian Deng, Sam Gross, Myle Ott, Marc'Aurelio Ranzato, and Arthur Szlam. 2021. Residual energy-based models for text. *J. Mach. Learn. Res.*, 22:40–1.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ben Buchanan, Andrew Lohn, Micah Musser, and Katerina Sedova. 2021. Truth, lies, and automation.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tomasz Darmetko. Fake or not? generating adversarial examples from language models.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liam Dugan, Daphne Ippolito, Arun Kirubarajan, and Chris 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*, pages 189–196, Online. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Anjalie Field, Su Lin Blodgett, Zeerak Waseem, and Yulia Tsvetkov. 2021. A survey of race, racism, and anti-racism in NLP. 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)*, pages 1905–1925, Online. Association for Computational Linguistics.

Leon Fröhling and Arkaitz Zubiaga. 2021. Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover. *PeerJ Computer Science*, 7:e443.

Rinaldo Gagiano, Maria Myung-Hee Kim, Xiuzhen Zhang, and Jennifer 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*, pages 119–127, Online. Australasian Language Technology Association.

Matthias Gallé, Jos Rozen, Germán Kruszewski, and Hady Elsahar. 2021. Unsupervised and distributional detection of machine-generated text. *arXiv preprint arXiv:2111.02878*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander 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*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Maurício Gruppi, Benjamin D Horne, and Sibel Adalı. 2022. Nela-gt-2021: A large multi-labelled news dataset for the study of misinformation in news articles. *arXiv preprint arXiv:2203.05659*.

Saurabh Gupta, Hong Huy Nguyen, Junichi Yamagishi, and Isao Echizen. 2020. Viable threat on news reading: Generating biased news using natural language models. In *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, pages 55–65, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas 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*, pages 1808–1822, Online. Association for Computational Linguistics.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Evan Lim Hong Jun, Chong Wen Haw, and Chieu Hai Leong. Robustness analysis of neural text detectors.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. *Advances in Neural Information Processing Systems*, 34:14542–14554.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language model for text generation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4492–4499. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

John Nash. 1953. Two-person cooperative games. *Econometrica*, 21(1):128–140.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. 2021. Generating fake cyber threat intelligence using transformer-based models. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Tal Schuster, Roei Schuster, Darsh J. Shah, and Regina Barzilay. 2020. The limitations of stylometry for detecting machine-generated fake news. *Computational Linguistics*, 46(2):499–510.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Adam Shostack. 2014. *Threat Modeling: Designing for Security*, 1st edition. Wiley Publishing.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Harald Stiff and Fredrik Johansson. 2021. Detecting computer-generated disinformation. *International Journal of Data Science and Analytics*, pages 1–21.

Joseph E Stiglitz et al. 1999. Knowledge as a global public good. *Global public goods: International cooperation in the 21st century*, 308:308–325.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395, Online. Association for Computational Linguistics.

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon 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*, pages 2001–2016, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chris J Vargo, Lei Guo, and Michelle A Amazeen. 2018. The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016. *New media & society*, 20(5):2028–2049.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Max Wolff and Stuart Wolff. 2020. Attacking neural text detectors. *arXiv preprint arXiv:2002.11768*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in Neural Information Processing Systems*, 32:9054–9065.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel

Configuration	Epochs	Batch size	Learning Rate	Max Seq. Len
Detection	10	64	1e-06	256

Configuration	prompt len	min len	max len	temp.	# beams	repet. pen.
Generation	10	20	None	1.0	3	1.3

Table 9: Detection and generation hyperparameters.

Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

A Appendix

A.1 Dataset Information

NELA GT 2020 The NELA GT 2020 dataset used to adapt the generators to the COVID domain is a collection of 699,803 articles from 493 sources in English collected from 2020-01-01 to 2020-12-31 about COVID. The text of the articles in this dataset was modified so that it cannot properly be used for news consumption but can still be used for text analysis. For articles with more than 200 tokens, 7 tokens are replaced with @ every 100 tokens. For articles with fewer than 200 tokens, 5 consecutive tokens are replaced with @ every 20 tokens. We pretrain the generators on this data and sample from the generator by disallowing the generation of @ signs.

Webtext and OpenAI Generations OpenAI released 250,000 documents from the OpenWebText dataset which was used to train GPT-2. They also released 250,000 outputs of each size of GPT-2. We use this data in our experiments. The validation and test splits are 5K documents each.

Generated Data We generate data using various sizes of GPT-2 and GPT-Neo using 15 words from NELA-GT COVID 2020 and 15 words from OpenWebText for GPT-3 as prompts. For GPT-3, the total number of generated documents is 50K. We use 40K for training and 5k each for validation and testing. The total number of training documents generated on COVID are listed in Table 10. In, addition, we generate 10K validation and testing splits for each decoding technique and model type.

B Hyperparameters

We performed hyperparameter tuning on the RoBERTa large model and used them for the other models. Due to high combinatorial number of configurations, we did not tune hyperparameters for each detector. We use Adam in our experiments.

Generator configuration	# Generated documents
GPT2-sm-covid top k=40	718662
GPT2-sm-covid nucleus p=0.96	845382
GPT2-sm-covid random	1073162
GPT2-md-covid top k=40	670062
GPT2-md-covid nucleus p=0.96	433662
GPT2-md-covid random	528022
GPT2-lg-covid top k=40	66360
GPT2-lg-covid nucleus p=0.96	101838
GPT2-lg-covid random	104190
GPT2-xl-covid top k=40	116798
GPT2-xl-covid nucleus p=0.96	220790
GPT2-xl-covid random	99194
GPT-Neo-covid top k=40	160078
GPT-Neo-covid nucleus p=0.96	172786
GPT-Neo-covid random	97380

Table 10: Number of documents generated on COVID for each generation hyperparameter configuration.

We use the default causal language modeling hyperparameters of the Huggingface toolkit to adapt the generators to the COVID domain. We maximize the batch size that fits on a GPU. For the larger models we used Deepspeed stage 3 and 4 A40 GPUs.

While previous work did not comment on generation hyperparameters, we found that generation quality is highly dependent on the hyperparameters of the generation process. In the COVID domain, after adapting the generators to the NELA GT 2020 dataset, we found the use of beam search to be necessary to obtain fluent text. In section 6, we discuss ideas which could help future work perform hyperparameter tuning more extensively and efficiently.

B.1 Detailed Results

For reference, we report the individual results presented throughout the paper without aggregations. On top of the accuracy metric, we also report F1 and equal error rate.

While in this work and in the literature it was observed that training on larger generators tends to generalize to smaller generators, we observe here a behavior that contradicts this trend. The RoBERTa large detector trained on GPT-3 outputs has relatively low performance on GPT-2 outputs (see Table 19). Future work should explore this inconsistency and establish if it is due to minor stylistic differences in the outputs of GPT-3 and GPT-3, and therefore to the general problem of shortcut learning, or to an inherent difference in the generated text from GPT-2 and GPT-3.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96			GPT-2 sm nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	69.23	65.77	34.00	60.85	42.69	43.36	78.56	80.98	27.24	78.61	81.51	28.21	78.58	79.62	24.08	74.77	70.21	31.04
GPT-2 md covid top k=40	50.08	29.45	49.95	70.20	61.63	36.04	81.22	83.73	26.13	80.00	82.91	27.63	79.46	80.63	23.70	75.94	71.98	29.78
GPT-2 md covid nucleus p=96	58.56	47.70	43.95	66.40	54.52	39.23	80.12	82.60	26.58	79.48	82.39	27.85	79.25	80.39	23.79	75.70	71.63	30.03
GPT-2 xl random	81.28	78.30	25.46	46.95	7.79	51.65	57.13	56.28	43.14	60.31	60.41	39.74	64.90	60.08	38.00	65.92	54.62	39.37
GPT-2 xl top k=40	48.88	5.09	50.58	94.61	94.88	9.59	78.80	82.29	29.35	79.11	82.45	28.92	84.39	85.47	20.05	84.11	82.97	19.93
GPT-2 xl nucleus p=96	55.10	25.27	47.16	65.57	54.79	39.46	74.86	78.29	31.10	75.48	78.76	30.53	77.57	77.70	22.75	76.39	72.41	29.52
GPT-2 lg random	85.05	83.39	20.78	46.72	7.02	51.77	59.94	60.28	40.23	64.10	65.49	36.95	69.22	66.63	33.36	71.01	63.85	34.95
GPT-2 lg top k=40	49.18	6.17	50.43	94.01	94.27	9.69	79.22	82.70	29.17	79.61	82.94	28.71	86.12	87.28	19.46	88.08	87.74	13.90
GPT-2 lg nucleus p=96	58.77	35.33	44.92	64.19	52.11	40.57	75.65	79.11	30.73	76.77	80.10	29.94	80.76	81.46	21.40	81.49	79.59	23.46
GPT-2 md random	93.89	93.82	7.08	45.40	2.47	52.45	59.18	59.22	40.84	64.75	66.33	36.51	81.02	81.76	21.30	77.91	74.64	27.82
GPT-2 md top k=40	49.75	8.19	50.13	93.02	93.26	9.87	79.06	82.55	29.24	79.54	82.88	28.74	87.69	88.88	18.96	88.45	88.17	13.29
GPT-2 md nucleus p=96	69.45	58.95	37.13	57.97	38.69	45.11	71.99	75.19	32.52	74.13	77.32	31.17	84.03	85.08	20.18	80.88	78.77	24.24
GPT-2 sm random	95.14	95.15	4.97	45.67	3.41	52.31	65.35	67.39	36.36	70.91	73.76	32.82	78.43	78.74	22.37	90.57	90.55	9.63
GPT-2 sm top k=40	53.54	20.61	48.07	91.63	91.81	10.13	79.40	82.88	29.10	79.89	83.22	28.60	86.80	87.98	19.24	94.86	95.06	8.49
GPT-2 sm nucleus p=96	76.64	71.38	30.52	58.57	40.09	44.70	74.90	78.33	31.08	77.11	80.44	29.78	81.31	82.09	21.19	92.86	93.00	8.82

Table 11: Performance of the detector based on BERT base for different training and testing combinations of hyperparameters and domains.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96			GPT-2 sm nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	70.20	65.88	33.88	64.06	52.88	40.46	77.37	78.00	24.09	76.92	74.54	27.32	62.79	47.61	41.90	64.58	49.26	40.91
GPT-2 md covid top k=40	52.58	31.96	48.40	77.26	74.58	27.49	83.18	84.51	21.71	84.45	84.16	16.77	69.26	60.34	36.72	58.80	35.63	44.89
GPT-2 md covid nucleus p=96	60.18	48.50	43.00	71.04	65.21	34.24	80.79	81.91	22.63	81.62	80.72	21.08	66.89	55.92	38.72	60.57	40.06	43.73
GPT-2 xl random	86.19	84.47	20.36	45.65	11.87	52.46	70.19	71.77	31.84	74.20	72.76	28.12	77.70	73.97	28.47	67.38	53.44	39.13
GPT-2 xl top k=40	51.63	11.07	49.15	91.94	92.53	13.82	81.62	84.29	26.41	88.03	88.97	17.51	86.45	85.65	17.22	67.57	53.84	38.98
GPT-2 xl nucleus p=96	59.51	34.98	44.58	71.74	67.80	32.54	79.32	81.97	27.35	84.69	85.45	18.57	82.18	80.23	23.12	65.64	49.71	40.43
GPT-2 lg random	90.90	90.28	13.70	45.27	10.70	52.67	73.84	76.05	29.88	79.56	79.57	20.48	84.52	83.26	19.99	74.40	66.79	33.27
GPT-2 lg top k=40	53.59	17.64	48.08	91.79	92.38	13.86	82.03	84.70	26.25	89.16	90.11	17.18	91.49	91.44	8.96	75.52	68.70	32.23
GPT-2 lg nucleus p=96	65.47	49.39	40.54	70.34	65.66	34.02	80.34	83.01	26.92	86.83	87.73	17.88	88.38	87.94	14.24	74.16	66.37	33.49
GPT-2 md random	96.21	96.17	4.72	43.01	3.46	53.84	75.34	77.73	29.14	82.71	83.25	19.26	92.64	92.68	7.87	84.65	82.42	22.36
GPT-2 md top k=40	57.69	30.01	45.71	91.51	92.10	13.92	81.83	84.50	26.33	88.90	89.85	17.26	94.26	94.38	7.62	80.33	76.30	27.36
GPT-2 md nucleus p=96	79.31	74.79	28.43	61.03	49.42	42.44	79.61	82.27	27.23	86.17	87.03	18.09	92.84	92.90	7.83	80.55	76.63	27.12
GPT-2 sm random	97.75	97.76	2.73	43.85	6.21	53.41	79.16	81.81	27.42	86.97	87.88	17.84	93.81	93.92	7.69	96.75	96.73	3.78
GPT-2 sm top k=40	76.05	69.61	31.70	91.22	91.81	13.99	82.23	84.89	26.17	89.57	90.52	17.06	95.14	95.29	7.50	97.52	97.52	2.67
GPT-2 sm nucleus p=96	90.50	89.81	14.31	62.75	52.71	41.05	81.09	83.77	26.62	88.45	89.40	17.39	93.84	93.95	7.68	96.81	96.79	3.66

Table 12: Performance of the detector based on BERT large for different training and testing combinations of hyperparameters and domains.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	68.81	58.02	37.57	74.88	67.64	32.81	85.35	84.04	19.63	80.32	76.70	26.87	63.24	44.61	42.08
GPT-2 md covid top k=40	59.62	37.97	44.34	81.91	78.63	25.59	89.98	89.61	12.68	85.54	83.87	20.54	63.21	44.53	42.11
GPT-2 md covid nucleus p=96	64.08	48.37	41.25	79.80	75.53	27.91	89.15	88.65	14.02	85.03	83.22	21.20	64.40	47.26	41.27
GPT-2 xl random	96.50	96.42	5.51	62.83	42.56	42.48	94.19	94.17	6.15	94.79	94.64	7.58	93.27	92.88	11.00
GPT-2 xl top k=40	78.14	72.46	30.08	98.56	98.57	1.86	95.85	95.90	5.29	95.97	95.90	5.46	84.56	82.01	23.08
GPT-2 xl nucleus p=96	83.17	80.06	24.71	87.80	86.40	18.67	94.70	94.71	5.41	94.99	94.86	7.23	86.80	85.01	20.29
GPT-2 lg random	97.67	97.64	3.35	63.99	45.33	41.69	95.26	95.29	5.35	96.59	96.56	4.31	96.44	96.36	5.60
GPT-2 lg top k=40	90.95	90.18	14.58	98.74	98.75	1.86	96.69	96.76	5.20	98.08	98.09	2.40	95.08	94.89	7.99
GPT-2 lg nucleus p=96	91.86	91.26	13.22	89.93	89.03	15.69	96.02	96.08	5.27	97.40	97.40	2.77	94.96	94.76	8.20
GPT-2 md random	98.70	98.70	1.36	55.69	23.03	46.92	95.70	95.75	5.30	97.36	97.35	2.85	98.31	98.30	2.10
GPT-2 md top k=40	96.97	96.91	4.65	98.65	98.66	1.86	96.81	96.88	5.19	98.28	98.29	2.39	98.43	98.43	1.87
GPT-2 md nucleus p=96	97.61	97.58	3.46	84.64	82.25	22.72	96.04	96.10	5.27	97.83	97.84	2.41	98.27	98.26	2.18
GPT-2 sm random	98.91	98.91	1.24	59.31	33.50	44.76	95.83	95.88	5.29	97.80	97.80	2.41	98.58	98.58	1.57
GPT-2 sm top k=40	98.42	98.41	1.91	98.50	98.51	1.87	96.71	96.78	5.20	98.48	98.49	2.38	98.93	98.93	1.26
GPT-2 sm nucleus p=96	98.19	98.18	2.35	86.46	84.67	20.43	95.70	95.75	5.30	98.13	98.14	2.39	98.63	98.63	1.48

Table 13: Performance of the detector based on ELECTRA large for different training and testing combinations of hyperparameters and domains.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96			GPT-2 sm nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	61.67	48.07	42.34	69.50	71.92	33.37	69.92	74.98	35.82	72.44	76.45	33.26	71.30	70.53	29.76	76.22	73.92	27.71
GPT-2 md covid top k=40	50.72	21.60	49.59	76.69	79.87	29.73	73.01	78.12	34.31	75.65	79.75	31.74	71.91	71.32	28.95	73.60	70.18	30.80
GPT-2 md covid nucleus p=96	55.18	33.43	46.86	73.73	76.72	31.13	71.42	76.53	35.07	74.13	78.21	32.44	71.97	71.41	28.86	75.95	73.55	28.04
GPT-2 xl random	79.00	75.31	27.67	40.67	27.80	56.88	56.02	60.96	45.20	61.31	64.44	40.38	72.96	71.47	29.21	70.55	65.31	34.22
GPT-2 xl top k=40	50.57	12.71	49.69	78.78	82.36	29.52	71.04	77.32	36.46	75.35	79.93	32.60	78.55	78.63	21.66	77.51	75.51	26.35
GPT-2 xl nucleus p=96	55.96	28.99	46.61	68.19	71.00	34.76	68.03	74.36	37.93	71.93	76.50	34.21	74.93	74.09	26.59	73.95	70.49	30.59
GPT-2 lg random	83.08	81.02	22.82	38.97	24.16	57.93	57.45	62.70	44.19	63.15	66.70	39.16	76.26	75.80	24.71	74.63	71.48	29.82
GPT-2 lg top k=40	51.53	15.84	49.17	78.85	82.43	29.50	71.29	77.56	36.34	75.88	80.45	32.37	82.05	82.72	20.25	81.69	80.93	20.64
GPT-2 lg nucleus p=96	59.28	37.68	44.52	67.38	70.04	35.24	68.56	74.89	37.66	73.08	77.67	33.65	78.65	78.75	21.62	78.22	76.47	25.44
GPT-2 md random	91.94	91.78	9.67	31.68	6.64	61.92	50.42	53.69	49.63	61.32	64.46	40.38	83.51	84.34	19.72	79.00	77.50	24.41
GPT-2 md top k=40	54.36	24.46	47.57	78.72	82.30	29.55	71.24	77.51	36.36	75.67	80.25	32.46	85.04	86.00	19.19	82.69	82.16	19.14
GPT-2 md nucleus p=96	70.46	61.40	36.08	57.88	57.62	42.22	64.73	70.94	39.68	70.30	74.79	35.03	83.72	84.57	19.64	80.41	79.32	22.48
GPT-2 sm random	93.68	93.66	6.55	33.47	11.26	61.02	57.99	63.35	43.82	66.90	71.07	36.88	84.77	85.71	19.28	88.47	88.79	13.58
GPT-2 sm top k=40	60.35	40.30	43.81	78.11	81.70	29.81	71.40	77.66	36.29	76.00	80.57	32.31	87.24	88.30	18.47	91.53	92.00	12.83
GPT-2 sm nucleus p=96	76.02	70.78	30.85	60.04	60.65	40.26	67.65	73.97	38.12	72.83	77.42	33.77	84.98	85.94	19.21	89.44	89.82	13.33

Table 14: Performance of the detector based on ELECTRA small for different training and testing combinations of hyperparameters and domains.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96			GPT-2 sm nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	78.58	76.33	25.99	54.06	17.68	47.84	71.86	63.04	35.20	79.00	75.03	27.99	64.68	48.89	40.93	56.05	25.64	46.67
GPT-2 md covid top k=40	55.38	33.70	46.74	58.78	31.89	45.09	72.27	63.78	34.84	78.53	74.33	28.50	57.68	31.83	45.63	54.01	19.40	47.84
GPT-2 md covid nucleus p=96	66.04	56.43	38.87	55.77	23.09	46.88	73.29	65.57	33.92	80.47	77.18	26.35	61.67	42.00	43.05	54.97	22.37	47.30
GPT-2 xl random	90.50	89.77	14.55	48.31	3.58	50.88	81.55	81.30	19.28	80.57	79.63	22.02	81.86	78.76	25.33	69.60	57.22	37.59
GPT-2 xl top k=40	53.60	17.11	48.09	97.28	97.35	5.04	90.58	91.25	14.82	91.43	91.93	13.16	87.62	86.42	18.02	61.50	38.85	43.39
GPT-2 xl nucleus p=96	62.76	42.83	42.48	62.41	44.48	42.46	88.70	89.32	15.32	88.79	89.18	13.81	84.53	82.44	22.11	63.05	42.72	42.37
GPT-2 lg random	94.68	94.52	7.80	48.34	3.69	50.86	85.47	85.83	16.26	86.07	86.19	14.55	89.36	88.55	15.52	80.04	75.51	28.07
GPT-2 lg top k=40	57.94	30.27	45.57	96.93	97.00	5.07	91.12	91.79	14.69	92.22	92.73	12.98	93.95	93.80	8.13	71.30	60.56	36.21
GPT-2 lg nucleus p=96	71.12	60.71	36.20	62.43	44.53	42.45	89.88	90.54	15.01	90.69	91.17	13.34	91.87	91.48	11.64	74.00	65.54	33.90
GPT-2 md random	97.70	97.70	2.38	47.73	1.43	51.17	87.98	88.56	15.52	88.48	88.85	13.89	95.25	95.19	5.80	89.66	88.65	16.32
GPT-2 md top k=40	68.01	54.55	38.69	96.22	96.28	5.14	91.08	91.75	14.70	92.25	92.76	12.97	96.95	96.97	3.52	80.00	75.45	28.11
GPT-2 md nucleus p=96	86.73	85.11	19.84	57.96	33.54	45.41	90.11	90.78	14.94	90.92	91.41	13.28	95.83	95.80	4.72	85.26	83.00	22.14
GPT-2 sm random	98.36	98.37	2.35	48.14	2.96	50.96	90.05	90.71	14.96	91.32	91.82	13.19	97.08	97.10	3.52	98.66	98.66	1.46
GPT-2 sm top k=40	93.91	93.68	9.12	96.62	96.68	5.10	91.38	92.05	14.62	92.49	93.00	12.92	97.92	97.95	3.46	98.77	98.77	1.45
GPT-2 sm nucleus p=96	96.18	96.12	5.11	59.66	37.90	44.32	90.89	91.56	14.74	92.05	92.56	13.02	97.18	97.20	3.51	98.59	98.59	1.46

Table 15: Performance of the detector based on RoBERTa base for different training and testing combinations of hyperparameters and domains.

Trained on →	GPT-2 xl random			GPT-2 xl top k=40			GPT-2 xl nucleus p=96			GPT-2 lg nucleus p=96			GPT-2 md nucleus p=96			GPT-2 sm nucleus p=96		
Tested on ↓	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER	Acc.	F1	EER
GPT-2 md covid random	67.61	55.48	38.60	73.73	65.14	34.10	68.16	55.01	38.54	71.71	62.05	35.62	58.31	31.94	45.32	52.76	14.26	48.55
GPT-2 md covid top k=40	61.76	42.84	42.93	83.41	80.50	24.28	79.13	74.47	28.66	79.50	75.10	28.20	58.39	32.17	45.27	51.91	11.34	49.00
GPT-2 md covid nucleus p=96	64.45	48.92	41.01	78.98	73.93	29.12	75.87	69.25	31.91	77.51	72.01	30.25	58.73	33.08	45.06	52.02	11.75	48.94
GPT-2 xl random	96.66	96.59	5.29	58.94	35.66	44.81	94.78	94.77	5.43	94.16	94.02	7.80	93.02	92.58	11.53	76.34	69.19	32.01
GPT-2 xl top k=40	75.97	68.84	32.18	96.93	96.98	4.71	96.51	96.56	4.84	96.08	96.06	4.29	86.18	84.16	21.18	58.12	28.51	45.56
GPT-2 xl nucleus p=96	81.77	78.02	26.32	86.04	84.65	19.50	95.91	95.95	4.89	95.40	95.35	5.56	86.92	85.13	20.24	62.71	40.97	42.68
GPT-2 lg random	98.15	98.14	2.51	60.45	39.46	43.83	96.32	96.37	4.85	96.74	96.75	3.50	97.17	97.12	4.44	89.15	87.89	17.59
GPT-2 lg top k=40	88.63	87.34	17.92	97.25	97.31	4.68	97.23	97.29	4.77	97.66	97.69	3.44	95.86	95.73	6.80	70.98	59.38	36.65
GPT-2 lg nucleus p=96	91.82	91.20	13.33	87.81	86.85	17.01	96.95	97.01	4.79	97.39	97.41	3.46	95.95	95.83	6.64	77.18	70.61	31.22
GPT-2 md random	99.04	99.04	1.16	52.59	17.50	48.60	96.86	96.92	4.80	97.27	97.29	3.47	98.88	98.88	1.18	96.73	96.64	5.76
GPT-2 md top k=40	96.34	96.25	5.87	97.25	97.31	4.68	97.32	97.38	4.76	97.93	97.96	3.42	98.66	98.66	1.61	86.08	83.91	21.57
GPT-2 md nucleus p=96	97.66	97.63	3.44	80.76	77.53	26.10	97.15	97.21	4.78	97.68	97.71	3.44	98.79	98.79	1.36	92.57	92.01	12.64
GPT-2 sm random	99.18	99.18	1.15	55.68	26.82	46.82	97.17	97.23	4.77	97.75	97.78	3.43	99.18	99.18	1.05	99.52	99.52	0.50
GPT-2 sm top k=40	98.88	98.88	1.16	97.23	97.29	4.68	97.45	97.51	4.75	98.06	98.09	3.41	99.26	99.26	1.05	99.51	99.51	0.52
GPT-2 sm nucleus p=96	98.75	98.75	1.34	83.93	81.91	22.27	97.27	97.33	4.77	97.95	97.98	3.42	99.14	99.14	1.06	99.49	99.49	0.56

Table 16: Performance of the detector based on RoBERTa large for different training and testing combinations of hyperparameters and domains.

	GPT-2 sm covid nucleus p=96	GPT-2 md covid nucleus p=96	GPT-2 lg covid nucleus p=96	GPT-2 xl covid nucleus p=96	GPT-Neo covid nucleus p=96
Roberta lg Acc.	91.69	75.87	90.52	68.95	79.16
Roberta lg F1	91.26	69.25	89.85	56.70	74.64
Roberta lg EER	12.35	31.91	14.35	37.92	28.57
Electra lg Acc.	95.24	89.15	94.38	78.56	74.27
Electra lg F1	95.33	88.65	94.43	74.72	68.23
Electra lg EER	5.98	14.02	6.10	28.06	32.44

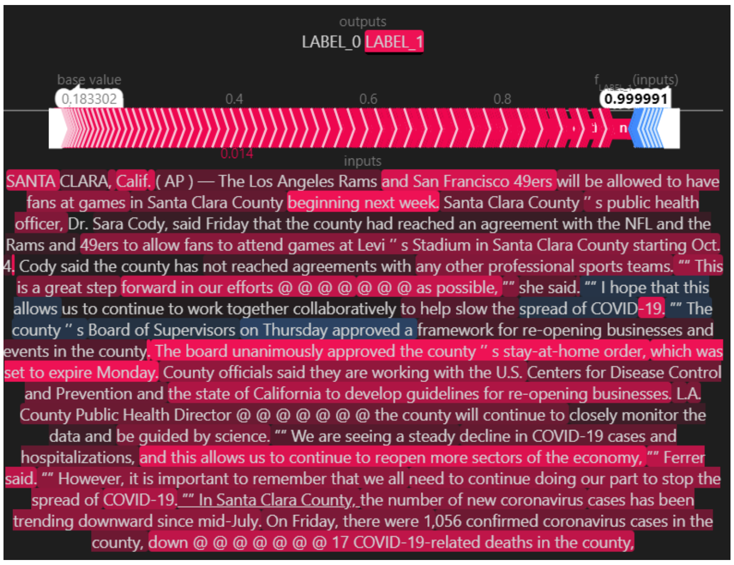
Table 17: Performance of detectors trained on the Webtext domain with examples of GPT-2 xl and tested on the COVID domain.

Detector	Trained on	Acc.	F1	EER
Roberta lg	GPT-2 xl random	58.64	30.86	45.21
Roberta lg	GPT-2 xl top k=40	59.86	38.02	44.22
Roberta lg	GPT-2 xl nucleus p=0.96	64.82	49.61	40.76
Roberta base	GPT-2 xl random	58.30	31.28	45.35
Roberta base	GPT-2 xl top k=40	54.97	25.29	47.23
Roberta base	GPT-2 xl nucleus p=0.96	63.55	54.80	40.23
Bert lg	GPT-2 xl random	57.95	30.74	45.55
Bert lg	GPT-2 xl top k=40	58.18	43.64	44.60
Bert base	GPT-2 xl random	59.28	36.63	44.59
Bert base	GPT-2 xl top k=40	55.85	33.54	46.50
Electra lg	GPT-2 xl random	57.00	26.17	46.19
Electra lg	GPT-2 xl top k=40	57.29	27.82	45.99
Electra sm	GPT-2 xl random	57.10	32.08	45.91
Electra sm	GPT-2 xl top k=40	55.23	53.71	45.09
Albert	GPT-2 xl random	60.76	51.50	42.21
Albert	GPT-2 xl top k=40	55.45	26.45	46.95

Table 18: Performance of various detectors trained on GPT-2 outputs and tested on GPT-3 with nucleus sampling $p = 0.96$

Tested on	Acc.	F1	EER
GPT-3 Davinci nucleus p=96	92.83	93.09	10.19
GPT-2 md covid random	54.35	67.35	47.58
GPT-2 md covid top k=40	52.59	65.67	48.53
GPT-2 md covid nucleus p=96	52.66	65.73	48.49
GPT-2 xl random	58.93	41.24	44.43
GPT-2 xl top k=40	56.03	34.36	46.37
GPT-2 xl nucleus p=96	52.55	25.29	48.53
GPT-2 lg random	61.69	47.27	42.44
GPT-2 lg top k=40	57.30	37.45	45.53
GPT-2 lg nucleus p=96	52.92	26.30	48.30
GPT-2 md random	78.78	76.35	26.12
GPT-2 md top k=40	61.48	46.82	42.60
GPT-2 md nucleus p=96	61.94	47.79	42.26
GPT-2 sm random	77.66	74.79	27.47
GPT-2 sm top k=40	61.81	47.52	42.35
GPT-2 sm nucleus p=96	62.88	49.73	41.54

Table 19: Performance of Roberta lg detector trained on outputs of GPT-3 with nucleus sampling and tested on outputs of GPT-2 xl.



SANTA CLARA, Calif. (AP) — The Los Angeles Rams and San Francisco 49ers will be allowed to have fans at games in Santa Clara County beginning next week. Santa Clara County ” s public health officer, Dr. Sara Cody, said Friday that the county had reached an agreement with the NFL and the Rams and 49ers to allow fans to attend games at Levi ” s Stadium in Santa Clara County starting Oct. 4. Cody said the county has not reached agreements with any other professional sports teams. “” This is a great step forward in our efforts @ @ @ @ @ @ @ as possible, “” she said. “” I hope that this allows us to continue to work together collaboratively to help slow the spread of COVID-19. “” The county ” s Board of Supervisors on Thursday approved a framework for re-opening businesses and events in the county. The board unanimously approved the county ” s stay-at-home order, which was set to expire Monday. County officials said they are working with the U.S. Centers for Disease Control and Prevention and the state of California to develop guidelines for re-opening businesses. L.A. County Public Health Director @ @ @ @ @ @ @ the county will continue to closely monitor the data and be guided by science. “” We are seeing a steady decline in COVID-19 cases and hospitalizations, and this allows us to continue to reopen more sectors of the economy, “” Ferrer said. “” However, it is important to remember that we all need to continue doing our part to stop the spread of COVID-19. “” In Santa Clara County, the number of new coronavirus cases has been trending downward since mid-July. On Friday, there were 1,056 confirmed coronavirus cases in the county, down @ @ @ @ @ @ @ 17 COVID-19-related deaths in the county.

Figure 2: Saliency map of the input text using SHAP. The most salient tokens according to SHAP are the punctuation tokens.