RAID: A Shared Benchmark for Robust Evaluation of Machine-Generated Text Detectors

Liam Dugan[1], **Alyssa Hwang**[1], **Filip Trhlik**[2], **Josh Magnus Ludan**[1]
Andrew Zhu[1], **Hainiu Xu**[3], **Daphne Ippolito**[4], **Chris Callison-Burch**[1]
University of Pennsylvania[1] University College London[2]
King's College London[3] Carnegie Mellon University[4]
{ldugan, ahwang16, jludan, andrz, ccb}@seas.upenn.edu
hainiu.xu@kcl.ac.uk, filip.trhlik.21@ucl.ac.uk, daphnei@cmu.edu

Abstract

Many commercial and open-source models claim to detect machine-generated text with extremely high accuracy (99% or more). However, very few of these detectors are evaluated on shared benchmark datasets and even when they are, the datasets used for evaluation are insufficiently challenging—lacking variations in sampling strategy, adversarial attacks, and open-source generative models. In this work we present RAID: the largest and most challenging benchmark dataset for machine-generated text detection. RAID includes over 6 million generations spanning 11 models, 8 domains, 11 adversarial attacks and 4 decoding strategies. Using RAID, we evaluate the out-of-domain and adversarial robustness of 8 open- and 4 closed-source detectors and find that current detectors are easily fooled by adversarial attacks, variations in sampling strategies, repetition penalties, and unseen generative models. We release our data[1] along with a leaderboard[2] to encourage future research.

1 Introduction

Large Language Models (LLMs) have been able to fool humans into thinking their outputs are human-written for roughly four years (Dugan et al., 2020; Clark et al., 2021). In that short span of time we have seen LLM-generated text be used for targeted phishing attacks (Baki et al., 2017; Hazell, 2023), mass spam and harassment (Weiss, 2019), disinformation campaigns (Sharevski et al., 2023; Spitale et al., 2023), and spurious scientific publication (Lund et al., 2023). In order to document and eventually mitigate such harms, we must develop robust automatic detectors of machine-generated text.

Many exciting and inventive methods have been proposed in recent years for detecting generated text (Crothers et al., 2023). However, when evaluating these methods, authors typically generate their own evaluation datasets and fail to test their models on shared resources—making it difficult to verify claims of accuracy and robustness. This has led to an erosion of trust in the efficacy of automatic detection methods and a generally fatalistic sentiment towards detection among researchers and practitioners (Sadasivan et al., 2023).

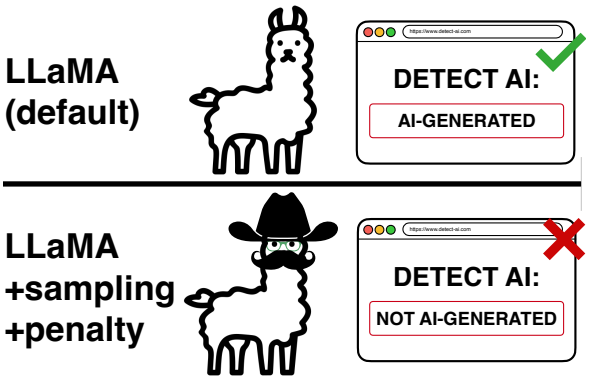

Figure 1: Detectors for machine-generated text are often highly performant on default model settings but fail to detect more unusual settings such as using random sampling with a repetition penalty.

To combat this trend, in this work, we introduce the Robust AI Detection (RAID) benchmark. RAID is the largest and most challenging benchmark of generated text ever released, consisting of 6M+ generations spanning 11 generators, 8 domains, 11 adversarial attacks, and 4 decoding strategies. Using RAID, we benchmark 12 detectors (8 open- and 4 closed-source). We find that detectors have difficulty generalizing to unseen models and domains and that simple changes such as changing the sampling strategy, adding a repetition penalty, and adversarially modifying text lead to marked decreases in performance.

2 Related Work

In Table 1 we show a comparison between RAID and other publicly available sources of generated

[1] https://github.com/liamdugan/raid
[2] https://raid-bench.xyz/leaderboard

Name	Size	Domain coverage?	Model coverage?	Sampling coverage?	Multilingual coverage?	Adversarial coverage?
TuringBench (Uchendu et al., 2021)	200k	✗	✓	✗	✗	✗
RuATD (Shamardina et al., 2022)	215k	✓	✓	✗	✗	✗
HC3 (Guo et al., 2023)	26.9k	✓	✗	✗	✓	✗
MGTBench (He et al., 2023)	2817	✓	✓	✗	✗	✓
MULTITuDE (Macko et al., 2023)	74.1k	✗	✓	✗	✓	✗
AuText2023 (Sarvazyan et al., 2023b)	160k	✓	✗	✗	✓	✗
M4 (Wang et al., 2023b)	122k	✓	✓	✗	✓	✗
CCD (Wang et al., 2023a)	467k	✗	✗	✗	✓	✓
IMDGSP (Mosca et al., 2023)	29k	✗	✓	✗	✗	✗
HC-Var (Xu et al., 2023)	145k	✓	✗	✗	✗	✗
HC3 Plus (Su et al., 2024)	210k	✓	✗	✗	✓	✗
MAGE (Li et al., 2024)	447k	✓	✓	✗	✗	✗
RAID (Ours)	6.2M	✓	✓	✓	✗	✓

Table 1: A comparison of the publicly available sources of generated text. Our provided dataset is the only one that contains a diverse selection of domains, sampling strategies, and adversarial attacks across recent generative models.

text. Among these, the most similar work to ours is Li et al. (2024), who create a dataset of 447k generations from 7 language model families across 10 domains to study detector robustness. Other resources typically focus on particular sub-areas such as multilingual text (Macko et al., 2023; Wang et al., 2023b), code (Wang et al., 2023a), question-answering (Guo et al., 2023; Xu et al., 2023; Su et al., 2024), and scientific papers (Mosca et al., 2023). Additionally, shared tasks such as AuTexTification (Sarvazyan et al., 2023b) and RuATD (Shamardina et al., 2022) have provided datasets and encouraged centralized evaluation and competition. While many shared resources do well at covering multiple generative models and domains, few include adversarial attacks and none include variation in decoding strategy—frequently even failing to list the strategy used. These datasets are insufficiently challenging and promote the inflated reports of detector accuracy.

Another way to evaluate robustness is through a small-scale comparative study. In these studies, one aspect of the generated text is varied and detector accuracy is compared across the variations. Such studies have shown that detectors lack robustness to unseen generative models (Stiff and Johansson, 2022; Pu et al., 2023b; Chakraborty et al., 2023), domains (Pagnoni et al., 2022; Pu et al., 2023a; Rodriguez et al., 2022), decoding strategies (Ippolito et al., 2020; Solaiman et al., 2019), prompts (Koike et al., 2023; Kumarage et al., 2023; Lu et al., 2023), repetition penalties (Fishchuk and Braun, 2023), and human edits (Gao et al., 2024).

Similar work specializing in adversarial robustness has shown that detectors are vulnerable to homoglyph attacks (Gagiano et al., 2021; Wolff, 2020; Macko et al., 2024), whitespace insertion (Cai and Cui, 2023), sentiment and factual alterations (Bhat and Parthasarathy, 2020), paraphrase attacks (Krishna et al., 2023; Sadasivan et al., 2023), and synonym replacement (Kulkarni et al., 2023; Pu et al., 2023a). Our work builds on this foundation and synthesizes many elements of the robustness literature into one singular systematic benchmark study.

3 Dataset Creation

3.1 Overview

In Figure 2, we illustrate the components of the RAID dataset. To create RAID, we first sample roughly 2,000 documents of human-written text from each of our 8 target domains (§3.2). For each document, we create a corresponding generation prompt using a template such as "Write a recipe for {title}" (§3.3). We then generate one output for each of our 11 models (§3.4), 4 decoding strategies (§3.5), and 11 adversarial attacks (§3.6). The RAID dataset consists of over 6M generations, the largest dataset of generated text to date.

3.2 Domains

Since different domains have been shown to induce LLMs to make diverse errors (Dugan et al., 2023), we prioritized domains that were both at high risk for abuse and were diverse and challenging. Our sources require factual knowledge (News, Wikipedia), generalization and reasoning (Abstracts, Recipes), creative and conversational skills (Reddit, Poetry), and knowledge of specific media (Books, Reviews). To avoid contamination, most of our human-written documents are taken

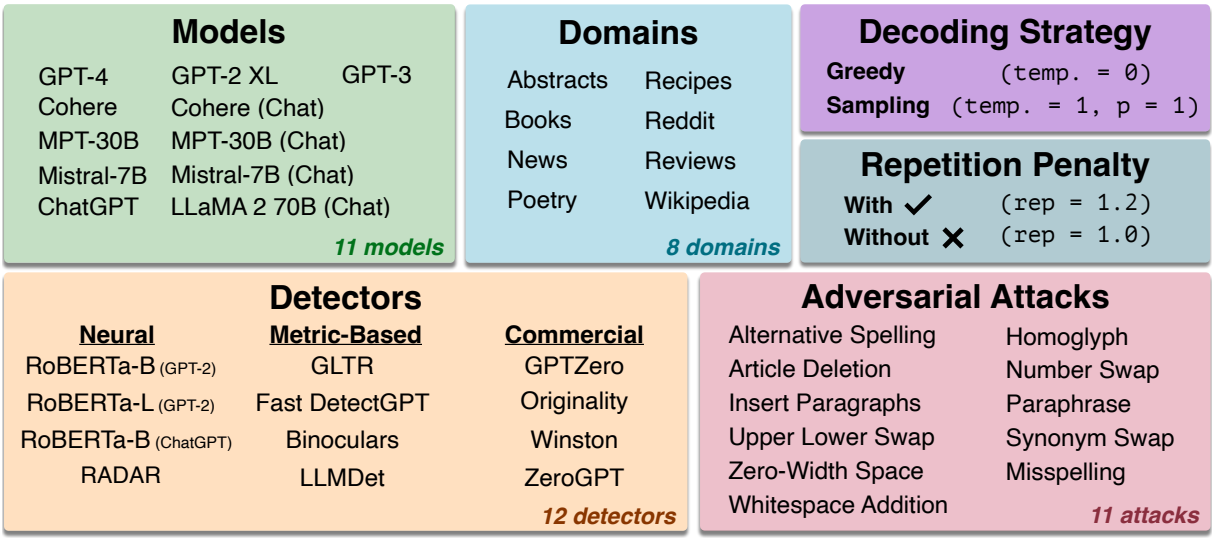

Figure 2: An overview of the structure of the RAID dataset. We generate 2,000 continuations for every combination of domain, model, decoding, penalty, and adversarial attack. This results in roughly 6.2 million generations for testing. We then evaluate each detector on all pieces of generated text in the dataset.

from publicly available pre-2022 datasets (see Appendix E.1).

3.3 Prompts

We prompt our generators in a zero-shot fashion using "Chat" templates for models fine-tuned on dialogue and "Non-Chat" templates for continuation models. Each prompt is nearly the same, with the exception of a "{title}" field that is dynamically replaced with the title of the corresponding human-written text (see Table 2). Unlike previous work (Verma et al., 2023; Xu et al., 2023), we intentionally avoid biasing the language model towards a particular length or generation style to better match our expectations of real-world scenarios. We engineered our prompts over multiple rounds to minimize degenerate repetition, unhelpful generation, and meta-commentary across all models (see Appendix E.3).

3.4 Models

We carefully chose a set of models that were maximally distinct from each other, offering us the widest range and variability of generated text. We focused on varying model sizes, open/closed source, and chat/completion style. Following work by Sarvazyan et al. (2023a), we select largest model from each model family and exclude third-party fine-tuned variants of base models in favor of the base models themselves. In total, we used four GPT models (GPT2, GPT3, GPT4, ChatGPT), three open-source models and their chat variants (Mistral 7B, MPT 30B, LLaMA 2 70B), and the Cohere *command* and *chat* model (see Appendix E.2).

Chat	Write the abstract for the academic paper titled "{title}".
Non-Chat	The following is the full text of the abstract for a research paper titled "{title}" from arxiv.org:

Table 2: The "Chat" and "Non-Chat" templates used for generation in the Abstracts domain. The "{title}" field is dynamically filled in with the title of the human-written document at generation time.

3.5 Decoding Strategies

The decoding strategy determines how tokens are selected from the language model's probability distribution. Previous work has shown that greedy decoding (i.e. selecting the most likely token at each time step) reduces the diversity of text and makes it easier to detect while sampling directly from the language model output distribution shows the opposite effect (Ippolito et al., 2020). Based on these findings, we generate two outputs per prompt, one with greedy decoding and the other with fully random sampling.

We also generate two additional outputs with Keskar et al. (2019)'s repetition penalty when available. This penalty works by down-weighting the probability of tokens that have previously appeared in the context window by some multiplicative factor θ, resulting in less repetitive output. We are the first to evaluate this penalty for detection at a

large scale and show that it significantly reduces the detectability of outputs. Following Keskar et al. (2019), we use $\theta = 1.2$ for our experiments.[3]

3.6 Adversarial Attacks

When selecting adversarial attacks, we assume that our adversary has exactly one query and no knowledge of the detector. Thus, we include the following 11 black-box, query-free attacks as opposed to gradient-based methods:

1. **Alternative Spelling**: Use British spelling
2. **Article Deletion**: Delete ('the', 'a', 'an')
3. **Add Paragraph**: Put `\n\n` between sentences
4. **Upper-Lower**: Swap the case of words
5. **Zero-Width Space**: Insert the zero-width space `U+200B` every other character
6. **Whitespace**: Add spaces between characters
7. **Homoglyph**: Swap characters for alternatives that look similar, e.g. e → e (`U+0435`)
8. **Number**: Randomly shuffle digits of numbers
9. **Misspelling**: Insert common misspellings
10. **Paraphrase**: Paraphrase with the fine-tuned T5-11B model from Krishna et al. (2023)
11. **Synonym**: Swap tokens with highly similar BERT (Devlin et al., 2019) candidate tokens

Following recommendations from Dyrmishi et al. (2023), we manually reviewed our data to ensure that adversarial attacks were inconspicuous. Thus, for each attack only a small percentage of the total available mutations were applied. For details on mutation percentages for each attack as well as other implementation details, see Appendix E.4.

3.7 Post-Processing

After all generations were completed, we removed prompts and left only the generated output. We then filtered out failed generations and balanced the dataset such that each human-written document has exactly one corresponding generation per model, decoding strategy, and adversarial attack.

4 Dataset

4.1 Statistics

The non-adversarial portion of the RAID dataset consists of 509,014 generations and 14,971 human-written documents for a total of 6,287,820 texts

[3]Only open-source models provide access to a repetition penalty. OpenAI and Cohere instead offer slightly different "frequency" and "presence" penalties (see Appendix E.5). Thus we vary repetition penalty only for open-source models.

Figure 3: Histogram of examples (y-axis) grouped by their repetitiveness measured via SelfBLEU score (x-axis). We see that both random sampling and repetition penalty greatly reduce repetitiveness for all models.

Model	Num. Gens	Toks	Self-BLEU	PPL -L7B	PPL -G2X
Human	14971	378.5	7.64	9.09	21.2
GPT 2	59884	384.7	23.9	8.33	8.10
GPT 3	29942	185.6	13.6	3.90	8.12
ChatGPT	29942	329.4	10.3	3.39	9.31
GPT 4	29942	350.8	9.42	5.01	13.4
Cohere	29942	301.9	11.0	5.67	23.7
(+ Chat)	29942	239.0	11.0	4.93	11.6
Mistral	59884	370.2	19.1	7.74	17.9
(+ Chat)	59884	287.7	9.16	4.31	10.3
MPT	59884	379.2	22.1	14.0	66.9
(+ Chat)	59884	219.2	5.39	7.06	56.3
LLaMA	59884	404.4	10.6	3.33	9.76
Total	509k	323.4	13.7	6.61	23.8

Table 3: Statistics for the generations in the base dataset without adversarial attacks. **PPL-L7B** refers to mean perplexity according to LLaMA 7B and **PPL-G2X** refers to mean perplexity according to GPT 2 XL.

when including adversarial attacks. On average, models are more repetitive than humans as measured by Self-BLEU (Zhu et al., 2018) and typically generate shorter passages (see Table 3). The mean perplexity is lower for models than humans, according to LLaMA 7B and GPT 2 XL.

4.2 Release Structure

To accompany the RAID dataset we also release an official public leaderboard[4] which will host the results from our analysis alongside other detector results submitted by public contributors. The leaderboard is split up into two sections—one for

[4]https://raid-bench.xyz/leaderboard

those who self-report having trained on the RAID dataset and one for those who do not. This is important to ensure that a clear distinction is made between detectors that are generalizing to out-of-domain data and those that are not.

To ensure fair competition, 10% of the RAID dataset is released without labels for use as the official hidden test set. We provide scripts to easily run detectors on this test set and calculate accuracy with respect to the hidden labels. Users can then submit their outputs to the leaderboard via a pull request (see Appendix D). We hope that this infrastructure encourages more comparison and shared evaluation of detectors.

4.3 RAID-extra

In addition to the over 6M+ generations in the core RAID train and test sets, we also release "RAID-extra", an additional dataset consisting of 2.3M generations from three extra domains not included in the main benchmark: Python Code, Czech News, and German News. In Appendix A we report evaluation results from our 12 detectors on RAID-extra and show that metric-based detectors perform surprisingly well on these types of unusual domains.

RAID-extra is the largest and most challenging dataset of generated code and multilingual text ever released. We hope it will be of value to the academic community.

5 Detectors

5.1 Detector Selection

We evaluate detectors from three categories: neural, metric-based, and commercial. Neural detectors typically involve fine-tuning a pre-trained language model such as RoBERTa (Liu et al., 2019) while metric-based detectors typically compute some metric using the output probabilities of an existing generative model. In contrast, commercial detectors tend to provide some documentation of their performance but disallow direct access to the models. We tested the following:

(i). **Neural**: RoBERTa-Base (GPT2), RoBERTa-Large (GPT2), RoBERTa-Base (ChatGPT), RADAR
(ii). **Metric-Based**: GLTR, Binoculars, Fast DetectGPT, LLMDet
(iii). **Commercial**: GPTZero, Originality, Winston, ZeroGPT

	τ=0.25	τ=0.5	τ=0.75	τ=0.95
R-B GPT2	8.71%	6.59%	5.18%	3.38%
R-L GPT2	6.14%	2.91%	1.46%	0.25%
R-B CGPT	21.6%	15.8%	15.1%	10.4%
RADAR	7.48%	3.48%	2.17%	1.23%
GLTR	100%	99.3%	21.0%	0.05%
F-DetectGPT	47.3%	23.2%	13.1%	1.70%
LLMDet	97.9%	96.0%	92.0%	75.3%
Binoculars	0.07%	0.00%	0.00%	0.00%
GPTZero	0.03%	0.00%	0.00%	0.00%
Originality	0.47%	0.25%	0.17%	0.07%
Winston	0.75%	0.55%	0.38%	0.21%
ZeroGPT	1.71%	1.42%	1.21%	0.90%

Table 4: False Positive Rates for detectors on RAID at naive choices of threshold (τ). We see that, for open-source detectors, thresholding naively results in unacceptably high false positive rates.

Unlike Li et al. (2024), we do not train our own neural models on our dataset because we wish to investigate the generalization ability of off-the-shelf detectors. For the metric-based detectors, we chose to use the default generative model in each repository to emulate the most realistic use-case.[5]

5.2 Detector Evaluation

Detectors work by taking in a sequence of tokens and outputting a scalar score. In order to convert this score to a binary prediction, we must select a scalar threshold τ such that if the score $s \geq \tau$ the sequence is predicted to be machine-generated.

In our work, we select a threshold for each model such that the resulting false positive rate of the detector is 5%. In practical terms, accuracy at a fixed FPR of 5% represents how well each detector identifies machine-generated text while only misclassifying 5% of human-written text. Our work is one of the first shared resources to fix and disclose FPR, following the rise of this evaluation paradigm in recent robustness research (Hans et al., 2024; Krishna et al., 2023; Soto et al., 2024).[6]

6 Findings

Finding 1: Default False Positive Rates (FPRs) of open-source detectors are dangerously high When applying a detector to a piece of text, it is important to decide on a threshold (τ) to use for binary classification. The principled way to determine this is to use a set of in-domain data to

[5]See Appendix F for more details on each detector tested
[6]See Appendix B for a discussion of why we chose this paradigm instead of the traditional precision/recall/F1 score

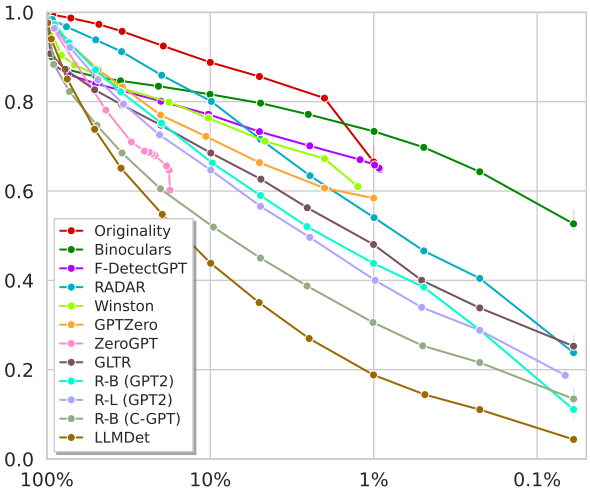

Figure 4: Detection accuracy (y-axis) vs. False Positive Rate (x-axis) for all detectors. We see that Binoculars works significantly better than other detectors at low FPR and that few detectors can operate at FPR<1%.

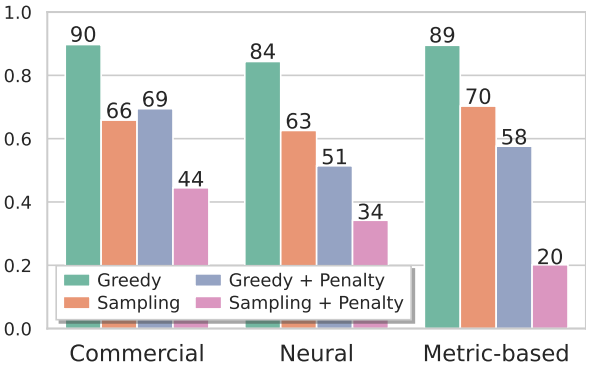

Figure 5: Accuracy at FPR=5% (y-axis) vs. decoding strategy across our three detector classes on non-adversarial text. We see that repetition penalty greatly reduces accuracy in both greedy decoding and sampling.

calibrate the threshold to a specific point along the receiver-operating curve. However, this is cumbersome and requires a set of readily accessible human-written data. Thus, in practice it is common to simply use some seemingly sensible default value (such as 0.5) for the threshold without further investigation.

In Table 4 we report the false positive rates of our detectors for various commonly chosen thresholds. We see that open-source detectors, especially metric-based detectors, exhibit dangerously high false positive rates when using these naive thresholds. On the contrary, closed-source detectors seem to be calibrated fairly well, with none having an FPR above 1.7%. Given this result, we advise practitioners to take care not to use naive-yet-sensible values for their detectors and instead calibrate detectors on in-domain data before using them.

Following this advice, for the remainder of the Findings section, we will be exclusively using thresholds that were calibrated to a FPR of 5% on the human-written portion of the RAID dataset (see Appendix C).

Finding 2: Detector accuracy varies substantially depending on target False Positive Rate In Figure 4 we report the results of an experiment where we varied the classification threshold and plotted detector accuracy vs. false positive rate. We found that our detectors were capable of achieving the high accuracies cited in many viral reports, but only at similarly high FPR. Some detectors failed to achieve the lowest FPR we tested, plateauing at 16.9% (ZeroGPT), 0.88% (FastDetectGPT), and 0.62% (Originality). Most detectors dropped steeply as FPR decreased from 100%, but Binoculars (Hans et al., 2024) was particularly strong at low FPR. Since detector accuracy varies so much with FPR, explicitly calibrating and reporting FPR is crucial for comparable, informative, and reproducible detection studies.

Finding 3: Repetition penalty drastically hurts accuracy for all detectors As shown in Figure 5, we observe a consistent pattern across detectors, detector categories, generators, and domains: adding a repetition penalty decreases accuracy by up to 32 points regardless of decoding strategy. The decoding strategy matters as well. Detectors in each category, across domains and generators, perform substantially better on greedy decoding than random sampling, even when taking repetition penalty into account.

This pattern is especially concerning because past studies have largely overlooked variations in decoding strategy when evaluating detectors. Furthermore, none have reported results on repetition penalty before our study. Since sampling with repetition penalty results in text that often sounds more human-like (Keskar et al., 2019), exposing these patterns is critical for reliable detection.

There are many possible penalties and decoding strategies one could use when generating text such as contrastive decoding (Li et al., 2023), eta and epsilon decoding (Hewitt et al., 2022), and typical sampling (Meister et al., 2023)—all of which are likely to reduce detector accuracy. We highly encourage robustness studies and open evaluations to investigate how well detectors can generalize to

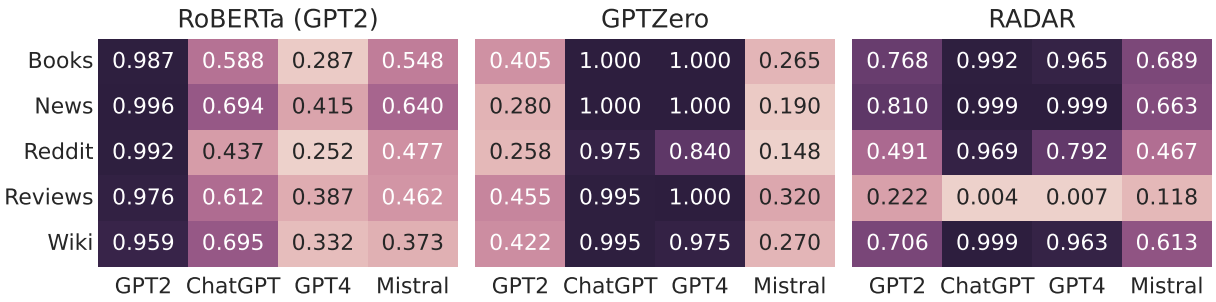

Figure 6: Heatmap measuring the accuracy of the RoBERTa-Large GPT2, GPTZero, and RADAR detectors across models and domains. We see a clear bias towards domains and models that the detectors have trained on.

	Open-Source								**Closed-Source**			
	Chat Models (llama-c, mistral-c, mpt-c)				**Non-Chat Models** (mistral, mpt, gpt2)				**Chat Models** (c-gpt, gpt4, cohere)		**Non-Chat Models** (cohere, gpt3)	
Dec. Strategy	greedy		sampling		greedy		sampling		greedy	sampling	greedy	sampling
Rep. Penalty?	✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗
R-B GPT2	84.1	52.3	77.9	26.2	98.6	44.1	60.5	35.4	70.9	41.7	65.1	52.5
R-L GPT2	79.7	41.1	71.4	19.5	98.5	43.0	67.2	53.4	61.4	34.7	61.1	48.6
R-B CGPT	80.2	63.3	75.0	39.3	53.3	26.4	14.9	1.7	59.1	38.1	46.5	39.0
RADAR	88.8	77.4	85.6	66.4	91.8	63.8	48.3	31.8	81.6	75.3	72.2	67.7
GLTR	89.8	67.5	83.9	38.3	99.6	56.9	44.5	0.5	80.7	54.3	75.6	63.7
F-DetectGPT	98.6	74.5	96.2	40.5	97.8	56.1	79.7	0.6	96.0	74.1	93.8	86.3
LLMDet	55.5	30.2	47.5	16.5	74.8	27.0	38.4	3.7	35.8	18.5	40.0	32.9
Binoculars	**99.9**	86.6	**99.7**	60.6	**99.9**	62.3	72.4	0.6	**99.2**	92.1	**99.0**	**95.0**
GPTZero	98.8	**93.7**	98.4	**82.5**	74.7	34.6	9.4	4.8	92.3	88.5	60.6	53.4
Originality	98.6	86.3	97.7	72.5	**99.9**	**64.1**	**89.0**	**51.2**	96.8	89.0	91.7	85.4
Winston	97.2	90.1	96.6	78.3	68.2	49.0	29.5	11.3	96.1	**93.7**	73.2	68.1
ZeroGPT(*)	95.4	80.7	90.5	54.9	85.1	57.2	16.0	0.3	92.1	65.8	83.4	72.7

Table 5: Accuracy Score at FPR=5% for all detectors across model groups and sampling strategies. Asterisks (*) indicate that the detector was unable to achieve the target FPR. We see that random sampling with a repetition penalty consistently makes output generations very difficult to detect, especially for open-source non-chat models.

alternative generation settings, especially if generative models are already trained on such outputs.

Finding 4: Seemingly strong, robust detectors can perform unexpectedly poorly The most accurate detectors—FastDetectGPT, Originality, Binoculars, etc.—may seem like reliable solutions to detection in general, but they sometimes deteriorate from perfect accuracy to complete failure (see Table 5). The changes in experimental settings were not particularly sophisticated either: simply changing the text generator, switching decoding strategies, or applying a repetition penalty was enough to introduce up to 95+% error rate. Our findings show that detectors tend not to generalize across different models or generation settings in the same domain. Compounded by the lack of evaluation at different false positive rates, domain-specific detectors for critical issues like fake news and education are particularly at risk of mislabeling human-written text as machine-generated without our awareness.

Finding 5: Detectors perform better on domains and models seen during training. Figure 6 shows the performance of RoBERTa-Large GPT2, GPTZero, and RADAR on a cross-section of models and domains from RAID. We see that RoBERTa GPT2 achieves 95+% accuracy on five domains generated by GPT2, but it rarely achieves beyond 60% accuracy on text of the same domain from different models. This detector is open-source, so we know that it was trained exclusively on GPT2 in an open-domain setting. We observe similar trends with RADAR, as it performs uncharacteristically poorly when detecting movie reviews regardless of generative model.

All detectors known to have constrained training data skew heavily towards test data with similar characteristics, leading us to believe that detectors perform better on domains and models seen during training. Some closed-source models such

	None	Paraphrase	Synonym	Misspelling	Homoglyph	Whitespace	Delete Articles
R-L GPT2	56.7	72.9 (+16.2)	79.4 (+22.7)	39.5 (-17.2)	21.3 (-35.4)	40.1 (-16.6)	33.2 (-23.5)
RADAR	70.9	67.3 (-3.6)	67.5 (-3.4)	69.5 (-1.4)	59.3 (-11.6)	66.1 (-4.8)	67.9 (-3.0)
GLTR	62.6	47.2 (-15.4)	31.2 (-31.4)	59.8 (-2.8)	24.3 (-38.3)	45.8 (-16.8)	52.1 (-10.5)
Binoculars	79.6	80.3 (+0.7)	43.5 (-36.1)	78.0 (-1.6)	37.7 (-41.9)	70.1 (-9.5)	74.3 (-5.3)
GPTZero	66.5	64.0 (-2.5)	61.0 (-5.5)	65.1 (-1.4)	66.2 (-0.3)	66.2 (-0.3)	61.0 (-5.5)
Originality	85.0	96.7 (+11.7)	96.5 (+11.5)	78.6 (-6.4)	9.3 (-75.7)	84.9 (-0.1)	71.4 (-13.6)

Table 6: Accuracy Score at FPR=5% for select detectors across different adversarial attacks. Colors indicate an increase, slight increase, slight decrease, and decrease in performance. We see that not all adversarial attacks affect models equally—with some occasionally even improving performance of detectors instead of harming them.

as GPTZero display similar behavior, allowing us to infer what data was used to train them. These findings demonstrate the need for multi-generator training corpora, especially since many publicly available neural detectors focus on only one or two generative models (Guo et al., 2023).

Finding 6: Different detectors are vulnerable to different types of adversarial attacks In Table 6, we see that Binoculars and other metric-based methods degrade as much as 36.1% when a small portion of words are swapped with synonyms. All detectors were sensitive to homoglyph attacks except for GPTZero which sustained only a 0.3% loss under the homoglyph attack while five others dropped an average of 40.6%. Detectors like RADAR that underwent adversarial training, unsurprisingly, were much more robust to adversarial attacks. These detector-dependent differences in vulnerability suggest that attacking an arbitrary detector without prior knowledge of the detector type or training distribution will be difficult. Adversaries may respond by attempting to discover what the detector was trained on—which our findings have shown could be possible—or attacking detectors with repeated queries.

In addition, we see that detector accuracy sometimes increases after an adversarial attack. RoBERTa GPT2, for example, improved after texts were paraphrased with T5 and after words were replaced with BERT-based synonyms. GPT2, RoBERTa, T5, and BERT are contemporaneous models trained on similar data, leading us to believe that detectors benefit from adversarial attacks that inadvertently modify text to be more similar to their training data. Our previous findings on the influence of training data on performance reinforce our hypothesis.

7 Conclusion

As the generation capabilities of language models have continued to increase, accurately and automatically detecting machine-generated text has become an important priority. Detection efforts have even surpassed the bounds of natural language processing research, spurring discussions by social media companies and governments on possibly mandating labels for machine-generated content. Despite the protective intentions of these mandates, our work shows that such regulations would be difficult to enforce even if they were implemented. Detectors are not yet robust enough for widespread deployment or high-stakes use: many detectors we tested are nearly inoperable at low false positive rates, fail to generalize to alternative decoding strategies or repetition penalties, show clear bias towards certain models and domains, and quickly degrade with simple black-box adversarial attacks.

The bulk of our findings may sound bleak, but we did uncover promising signs of improvement. Binoculars, for example, performed impressively well across models even at extremely low false positive rates, Originality achieved high precision in some constrained scenarios, and GPTZero was unusually robust to adversarial attacks. We believe that openly evaluating detectors on large, diverse, shared resources is critical to accelerating progress—and trust—in detection. Evaluating robustness is particularly important for detection, and it only increases in importance and the scale of public deployment grows.

We also need to remember that detection is just one tool for a larger, even more valuable motivation: preventing harm by the mass distribution of text. Detecting machine-generated text was a useful proxy for identifying harmful text for a long time, but language models have improved to the point that generated text is frequently legitimate and not harmful (Schuster et al., 2020). Therefore,

detecting specific harmful elements—like misinformation, hate speech, and abuse—should take precedence over whether or not the text was authored by a machine. Knowing if a text was machine-generated, however, does still offer insights on the types of errors we can expect or the recency of the facts cited within. We hope that our analyses and the RAID dataset are a step toward a future in which AI detection tools are safely integrated into society as a multi-pronged approach to reducing harm. We encourage future work to build on this by including more models, languages, and generation settings in future shared resources.

Limitations

While we attempt to cover a wide variety of domains, models, decoding strategies and adversarial attacks in our dataset, we recognize that there can never be a truly comprehensive dataset for robustness. In particular, our dataset lacks the inclusion of multilingual text in many diverse domains. We release our RAID-extra data to help begin this process but we acknowledge the limited nature of this approach (only having multilingual text in the news domain). We encourage future work to expand on our foundation and use our tools to create truly robust shared benchmarks in many languages.

Furthermore, as the state-of-the-art in language modeling continues to improve, datasets of generated text will naturally obsolesce and will need to be continually maintained with new generations. This creates issues with shared evaluations as detectors will need to be re-run on any new dataset items and any accuracy metrics will have to be updated. While we believe this dataset will continue to be useful for many years, we do acknowledge this limitation and plan to alleviate this by occasionally releasing new updated versions.

Finally, and most importantly, the concept of a public benchmark for out-of-domain robustness is an inherently limited one. As practitioners seek to improve performance on our benchmark they will undoubtedly specialize to the particular aspects of robustness we cover. This will lead to overfitting, even if detectors are not explicitly trained on examples. Such overfitting will result in the reappearance of exactly the problems we wished to alleviate by creating this dataset, namely that detector accuracies are generally over-reported. We trust that this process will, to some extent, be alleviated by regular releases of new versions and keeping a set of hidden test data private. That being said, it does not nullify the utility of the dataset as a resource for profiling robustness of classifiers.

Ethics Statement

Detecting generated text is often accusatory in nature and can frequently result in disciplinary or punitive action taken against the accused party. This can cause significant harm even when detectors are correct, but especially when they are incorrect. This is especially problematic given recent work by Liang et al. (2023c) showing that detectors are biased against non-native English writers. Our results also support this and suggest that the problem of false positives remains unsolved.

For this reason, we are opposed to the use of detectors in any sort of disciplinary or punitive context and it is our view that poorly calibrated detectors cause more harm than they solve. Therefore, until better evaluation standards are widely adopted in the detection community, the use of detectors in this fashion should be discouraged. We intend for our work to be the start of this conversation and look forward to a future where machine-generated detectors are deployed in safe and responsible ways.

Acknowledgements

The authors would like to thank the members of the lab of Chris Callison-Burch for their testing and detailed feedback on the contents of this paper. In addition, we'd like to thank Professor Dan Roth for his early and enthusiastic support of the project.

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200005. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Aaditya Bhat. 2023. Gpt-wiki-intro.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru,

Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models.

Micael Arman. 2020. Poems dataset (nlp). https://www.kaggle.com/datasets/michaelarman/poemsdataset.

Shahryar Baki, Rakesh Verma, Arjun Mukherjee, and Omprakash Gnawali. 2017. Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 469–482, New York, NY, USA. Association for Computing Machinery.

David Bamman and Noah A. Smith. 2013. New alignment methods for discriminative book summarization.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2023. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature.

Meghana Moorthy Bhat and Srinivasan 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*, pages 48–53, Online. Association for Computational Linguistics.

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Matyáš Boháček, Michal Bravanský, Filip Trhlík, and Václav Moravec. 2022. Fine-grained czech news article dataset: An interdisciplinary approach to trustworthiness analysis.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Shuyang Cai and Wanyun Cui. 2023. Evade chatgpt detectors via a single space.

Megha Chakraborty, S.M Towhidul Islam Tonmoy, S M Mehedi Zaman, Shreya Gautam, Tanay Kumar, Krish Sharma, Niyar Barman, Chandan Gupta, Vinija Jain, Aman Chadha, Amit Sheth, and Amitava Das. 2023. Counter Turing test (CT2): AI-generated text detection is not as easy as you may think - introducing AI detectability index (ADI). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2206–2239, Singapore. Association for Computational Linguistics.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah 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)*, pages 7282–7296, Online. Association for Computational Linguistics.

NLP Team Cohere. 2024. World-class ai, at your command. Accessed: 2024-02-02.

Evan N. Crothers, Nathalie Japkowicz, and Herna L. Viktor. 2023. Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE Access*, 11:70977–71002.

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.

Liam Dugan, Daphne Ippolito, Arun Kirubarajan, Sherry Shi, and Chris Callison-Burch. 2023. Real or fake text? investigating human ability to detect boundaries between human-written and machine-generated text. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Salijona Dyrmishi, Salah Ghamizi, and Maxime Cordy. 2023. How do humans perceive adversarial text? a reality check on the validity and naturalness of word-based adversarial attacks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8822–8836, Toronto, Canada. Association for Computational Linguistics.

Vitalii Fishchuk and Daniel Braun. 2023. Efficient black-box adversarial attacks on neural text detectors.

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.

Chujie Gao, Dongping Chen, Qihui Zhang, Yue Huang, Yao Wan, and Lichao Sun. 2024. Llm-as-a-coauthor: The challenges of detecting llm-human mixcase.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers.

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.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Jesus Guerrero, Gongbo Liang, and Izzat Alsmadi. 2022. A mutation-based text generation for adversarial machine learning applications.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text.

Julian Hazell. 2023. Large language models can be used to effectively scale spear phishing campaigns. *arXiv preprint arXiv:2305.06972*.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. Mgtbench: Benchmarking machine-generated text detection.

John Hewitt, Christopher Manning, and Percy Liang. 2022. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *Advances in Neural Information Processing Systems*.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2023. How you prompt matters! even task-oriented constraints in instructions affect llm-generated text detection.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense.

Pranav Kulkarni, Ziqing Ji, Yan Xu, Marko Neskovic, and Kevin Nolan. 2023. Exploring semantic perturbations on grover.

Tharindu Kumarage, Paras Sheth, Raha Moraffah, Joshua Garland, and Huan Liu. 2023. How reliable are ai-generated-text detectors? an assessment framework using evasive soft prompts.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training

data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. Mage: Machine-generated text detection in the wild.

Gongbo Liang, Jesus Guerrero, and Izzat Alsmadi. 2023a. Mutation-based adversarial attacks on neural text detectors.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023b. Holistic evaluation of language models.

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023c. Gpt detectors are biased against non-native english writers.

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.

Ning Lu, Shengcai Liu, Rui He, Qi Wang, Yew-Soon Ong, and Ke Tang. 2023. Large language models can be guided to evade ai-generated text detection.

Brady D Lund, Ting Wang, Nishith Reddy Mannuru, Bing Nie, Somipam Shimray, and Ziang Wang. 2023. Chatgpt and a new academic reality: Artificial intelligence-written research papers and the ethics of the large language models in scholarly publishing. *Journal of the Association for Information Science and Technology*, 74(5):570–581.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Dominik Macko, Robert Moro, Adaku Uchendu, Jason Lucas, Michiharu Yamashita, Matúš Pikuliak, Ivan Srba, Thai Le, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2023. MULTITuDE: Large-scale multilingual machine-generated text detection benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9960–9987, Singapore. Association for Computational Linguistics.

Dominik Macko, Robert Moro, Adaku Uchendu, Ivan Srba, Jason Samuel Lucas, Michiharu Yamashita, Nafis Irtiza Tripto, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2024. Authorship obfuscation in multilingual machine-generated text detection.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

NLP Team MosaicML. 2023. Introducing mpt-30b: Raising the bar for open-source foundation models. Accessed: 2023-06-22.

Edoardo Mosca, Mohamed Hesham Ibrahim Abdalla, Paolo Basso, Margherita Musumeci, and Georg Groh. 2023. Distinguishing fact from fiction: A benchmark dataset for identifying machine-generated scientific papers in the LLM era. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 190–207, Toronto, Canada. Association for Computational Linguistics.

OpenAI. 2022. ChatGPT: Optimizing Language Models for Dialogue.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Artidoro Pagnoni, Martin Graciarena, and Yulia Tsvetkov. 2022. Threat scenarios and best practices to detect neural fake news. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1233–1249, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Sayak Paul and Soumik Rakshit. 2021. arxiv paper abstracts. https://www.kaggle.com/datasets/spsayakpaul/arxiv-paper-abstracts.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only.

J. Pu, Z. Sarwar, S. Abdullah, A. Rehman, Y. Kim, P. Bhattacharya, M. Javed, and B. Viswanath. 2023a. Deepfake text detection: Limitations and opportunities. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1613–1630, Los Alamitos, CA, USA. IEEE Computer Society.

Xiao Pu, Jingyu Zhang, Xiaochuang Han, Yulia Tsvetkov, and Tianxing He. 2023b. On the zero-shot generalization of machine-generated text detectors.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Veselin Raychev, Pavol Bielik, and Martin Vechev. 2016. Probabilistic model for code with decision trees. *SIGPLAN Not.*, 51(10):731–747.

Juan Diego Rodriguez, Todd Hay, David Gros, Zain Shamsi, and Ravi Srinivasan. 2022. Cross-domain detection of GPT-2-generated technical text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1213–1233, Seattle, United States. Association for Computational Linguistics.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected?

Areg Mikael Sarvazyan, José Ángel González, Paolo Rosso, and Marc Franco-Salvador. 2023a. Supervised machine-generated text detectors: Family and scale matters. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 14th International Conference of the CLEF Association, CLEF 2023, Thessaloniki, Greece, September 18–21, 2023, Proceedings*, page 121–132, Berlin, Heidelberg. Springer-Verlag.

Areg Mikael Sarvazyan, José Ángel González, Marc Franco-Salvador, Francisco Rangel, Berta Chulvi, and Paolo Rosso. 2023b. Overview of autextification at iberlef 2023: Detection and attribution of machine-generated text in multiple domains.

Dietmar Schabus, Marcin Skowron, and Martin Trapp. 2017. One million posts: A data set of german online discussions. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1241–1244, New York, NY, USA. Association for Computing Machinery.

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.

Tatiana Shamardina, Vladislav Mikhailov, Daniil Chernianskii, Alena Fenogenova, Marat Saidov, Anastasiya Valeeva, Tatiana Shavrina, Ivan Smurov, Elena Tutubalina, and Ekaterina Artemova. 2022. Findings of the the ruatd shared task 2022 on artificial text detection in russian. In *Computational Linguistics and Intellectual Technologies*. RSUH.

Filipo Sharevski, Jennifer Vander Loop, Peter Jachim, Amy Devine, and Emma Pieroni. 2023. Talking abortion (mis) information with chatgpt on tiktok. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 594–608. IEEE.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models.

Rafael Rivera Soto, Kailin Koch, Aleem Khan, Barry Chen, Marcus Bishop, and Nicholas Andrews. 2024. Few-shot detection of machine-generated text using style representations.

Giovanni Spitale, Nikola Biller-Andorno, and Federico Germani. 2023. Ai model gpt-3 (dis)informs us better than humans. *Science Advances*, 9(26):eadh1850.

Harald Stiff and Fredrik Johansson. 2022. Detecting computer-generated disinformation. *International Journal of Data Science and Analytics*, 13:363–383.

Zhenpeng Su, Xing Wu, Wei Zhou, Guangyuan Ma, and Songlin Hu. 2024. Hc3 plus: A semantic-invariant human chatgpt comparison corpus.

Edward Tian and Alexander Cui. 2023. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura,

Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2023. Ghostbuster: Detecting text ghostwritten by large language models.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark. Association for Computational Linguistics.

Jian Wang, Shangqing Liu, Xiaofei Xie, and Yi Li. 2023a. Evaluating aigc detectors on code content.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, and Preslav Nakov. 2023b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection.

Max Weiss. 2019. Deepfake bot submissions to federal public comment websites cannot be distinguished from human submissions. *Technology Science*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Max Wolff. 2020. Attacking neural text detectors.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. LLMDet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2113–2133, Singapore. Association for Computational Linguistics.

Han Xu, Jie Ren, Pengfei He, Shenglai Zeng, Yingqian Cui, Amy Liu, Hui Liu, and Jiliang Tang. 2023. On the generalization of training-based chatgpt detection methods.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models.

A Experiments on Multilingual and Code Generations (RAID-Extra)

In addition to the main RAID dataset we also release RAID-Extra: a collection of 2.3M generations in three extra challenging domains: Python Code, Czech News, and German News. These extra experiments were not included in the main benchmark as we felt that they were out of scope for most detectors and should not be used as a basis for comparison. Nonetheless, we were still curious to see what sorts of insights they can give us on detector performance.

A.1 Data Generation

Following Macko et al. (2023), multilingual prompts were written in the target language by a native speaker rather than being written in English and explicitly requesting that the model complete the generation in the target language. We found this to be the most effective method to get our generative models to adhere to the target language.

For Python Code generations, we applied an additional post-processing step as, in this domain, generative models had a tendency to write code between sets of triple backtick characters (```) and give natural language explanations of the code outside of the backticks. Thus for this domain and this domain only, we extracted the text between these sets of backticks and discarded all others. This was done to ensure that detectors could not use text descriptions of code for detection and would instead have to rely on the code itself.

A.2 Results

In Table 7 we report the accuracies of our 12 detectors on generations from RAID-extra at 5% FPR. We see an interesting trend, that being the relatively strong performance of metric-based classifiers as compared to neural and commercial detectors. We suspect that metric-based classifiers are particularly well suited for such rare domains as they can be given any generative model to calculate their probabilities.

In Figure 7 we show a heatmap of the performance of our detectors across the extra domains from select models. We see that Binoculars performs decently well when detecting Czech

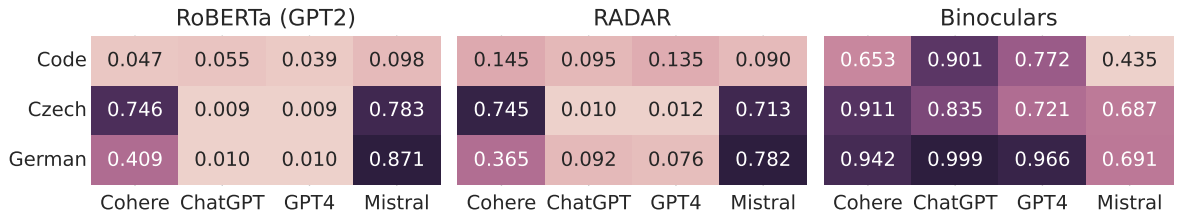

Figure 7: Heatmaps of accuracy for three of our detectors on German News, Python Code, and Czech News generations. We see that metric-based detectors have an edge over neural detectors in their ability to generalize to these unusual domains.

	Code	Czech	German	Total
R-B GPT2	13.4	48.4	39.7	38.2
R-L GPT2	12.7	53.1	48.4	43.5
R-B CGPT	24.0	38.7	51.5	41.1
RADAR	12.9	51.1	53.2	44.7
GLTR	40.7	51.9	68.9	56.7
F-DetectGPT	51.1	55.2	75.5	62.7
LLMDet	17.5	24.0	10.6	17.3
Binoculars	**59.9**	67.0	76.7	**69.6**
GPTZero	33.8	33.6	49.5	39.0
Originality	8.5	69.8	**89.1**	55.8
Winston	24.5	**70.3**	73.8	56.2
ZeroGPT	13.8	49.3	51.7	38.3

Table 7: Accuracy of our 12 detectors at FPR=5% on RAID-extra domains (Python Code, Czech News, and German News). We see that metric based detectors generally perform better than neural detectors.

news articles despite the underlying generative model, Falcon 7B (Almazrouei et al., 2023) being trained with five times as much German data as Czech data (Penedo et al., 2023). This seems to suggest that strong metric-based detectors for low-resource languages can be bootstrapped from highly-multilingual language models. Future work is necessary to understand the optimal setup in such scenarios.

B Fixed FPR Accuracy vs. F1 Score

Throughout our work we report accuracy on machine generated text at a set FPR because we believe it is the most intuitive way of understanding the performance of models in high-risk scenarios (i.e. "What percentage of generations are detected given that we tolerate an x% chance of wrongly accusing someone"). Reporting the more standard F1 score is not only less intuitive but also treats false positives and false negatives as equivalent—which is not the case when dealing with high-risk scenarios or ones where detectors are repeatedly applied to texts from the same author.

In addition, since our dataset has a roughly 40:1 ratio of generated to human-written text, precision scores will artificially favor true positives over false positives as most all examples in the dataset are positive examples. However, in the real world, this ratio is reversed and the majority of texts are human written. Thus precision scores systematically over-represent the capabilities of detectors when used as a metric on a dataset like ours. We hope that our work can help to shed light on this issue and how easy it is to accidentally over-represent the performance of classifiers.

C Per-Domain Threshold Tuning

When tuning the False Positive Rate of classifiers to a specific percentage (in our case 5%), it is important to look not just at total FPR across all human texts, but also at FPR for each individual domain in the dataset. In our work, we ensure that classification thresholds are determined on a per-domain basis, i.e. that the FPRs of every detector on every domain of the data should be 5%. While this undoubtedly adds complexity to the evaluation, it is an important step to ensure that detectors are being evaluated fairly with respect to one another (see Appendix F.2 for details about the threshold searching procedure).

To drive home the importance of this point, in Table 8 we show the FPR of each classifier at a 5% total FPR threshold broken up by domain. As we can see, while the total FPR is consistently 5%, many detectors have particularly acute domain-specific weaknesses: RADAR has a 20.4% FPR on Reviews, GLTR has a 33.4% FPR on Recipes, and Originality has a 13% FPR on Wikipedia.

This asymmetric variation of FPR creates a dampening effect whereby the inclusion of weaker, more obscure domains reduces the accuracy of a classifier on more common domains—ultimately lowering total accuracy in the process.

In order to avoid this issue, we ensure that our thresholds are chosen on a per-domain basis. That

	τ	News	Wiki	Reddit	Books	Abstracts	Reviews	Poetry	Recipes	Total
R-B GPT2	0.759	1.0%	3.2%	3.7%	4.0%	1.6%	3.1%	15.4%	7.3%	5.0%
R-L GPT2	0.307	1.8%	7.0%	2.4%	2.5%	1.3%	4.0%	15.4%	5.1%	5.0%
R-B C-GPT	0.988	4.4%	4.0%	0.7%	9.0%	0.0%	1.1%	0.5%	18.6%	5.0%
RADAR	0.343	0.2%	1.2%	10.7%	2.0%	4.2%	20.4%	8.6%	0.1%	5.0%
GLTR	0.818	0.4%	0.8%	1.1%	0.0%	0.2%	0.1%	1.8%	33.4%	5.0%
F-DetectGPT	0.920	5.1%	1.6%	1.9%	6.3%	1.8%	2.8%	7.5%	2.2%	3.7%
LLMDet	1.000	10.2%	12.8%	6.1%	2.7%	0.2%	3.9%	3.4%	0.1%	5.0%
Binoculars	0.090	2.8%	5.7%	7.2%	4.0%	5.3%	5.7%	3.8%	5.9%	5.0%
GPTZero	0.068	3.0%	2.0%	3.0%	13.0%	10.0%	5.0%	0.0%	1.0%	4.6%
Originality	0.494	4.0%	13.0%	2.0%	3.0%	4.0%	4.0%	3.0%	7.0%	5.0%
Winston	0.892	0.0%	10.0%	0.0%	13.0%	0.0%	0.0%	4.0%	13.0%	5.0%
ZeroGPT	1.000	29.0%	48.0%	0.0%	1.0%	0.0%	5.0%	0.0%	52.0%	16.9%

Table 8: False Positive Rates of our detectors on the RAID dataset broken up by domain when naively using a single threshold (τ). We see that while the total FPR across all domains still sums to 5%, individual domains see substantial fluctuation in FPR values.

is, we find the threshold for each detector for each domain that results in 5% FPR on that domain (see Table 13).

D Leaderboard and Pypi Package

In order to truly achieve our goal of standardizing detector evaluation, it is important for RAID to not only be sufficiently challenging, but also have a simple, straightforward interface for submission and comparison. As discussed in section 4.2, we solve this problem by releasing a shared leaderboard and a Pypi package to make submitting to the leaderboard easy. The RAID package can be installed by running:

```
$ pip install raid-bench
```

In Figure 9 we show how to use our pypi package to load the RAID test set and run a detector on the texts. After getting the `predictions.json` file, submitting to the leaderboard simply involves making a folder in the repository, filling out metadata, and creating a pull request.

In Figure 8 we show a screenshot of the leaderboard page on our project website. Submissions are open to the public and are automatically accepted and evaluated via pull requests to our git repository. The data used for evaluation on the leaderboard is the 10% test split of the core RAID dataset that is released without labels.

E Dataset Details

E.1 Domains

In Table 9 we report the exact number of documents sampled from each domain. The only two datasets that include post-2021 documents are Wikipedia and Abstracts. The Reviews domain did not have 2,000 documents and so we sampled the maximum amount. For each domain, we provide a detailed list of all human-written sources with metadata along with the RAID dataset in our code repository. A detailed description of the contents of each domain is as follows:

Book summaries (Bamman and Smith, 2013) This dataset contains plot-centric summaries of books along with their titles. We chose this dataset due to the first-person narrative style and because we expect generators and detectors with knowledge of the source material to have an advantage.

BBC News Articles (Greene and Cunningham, 2006) This dataset contains BBC articles with associated titles. The articles are spread out evenly across 5 categories (sport, technology, entertainment, politics, and business). This dataset was chosen since good generation requires factuality and because News is a large area for LLM-based harm.

Poems (Arman, 2020) This dataset contains poems collected from `poemhunter.com` with their titles and genre. The poems are randomly spread out over genres and topics. We hypothesize that LLMs will write generic and repetitive poetry and that this tendency should be detectable.

Recipes (Bień et al., 2020) This dataset consists of recipes and their dish names. Recipes are a combination of a list of ingredients and a numbered list of steps. This dataset is difficult because it requires significant common sense reasoning, which

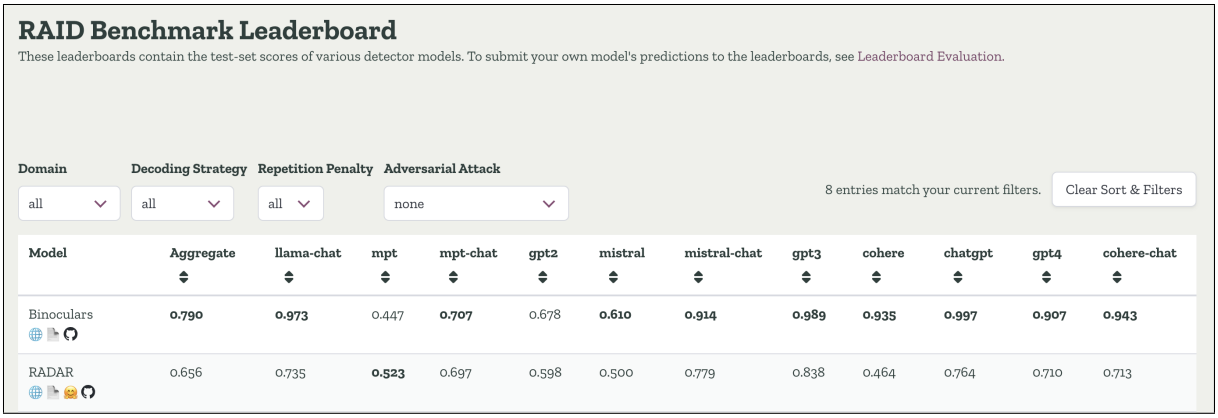

Figure 8: Screenshot of the RAID leaderboard accessible at https://raid-bench.xyz/leaderboard

```python
from raid import run_detection
from raid.utils import load_data

# Define your detector function
def my_detector(texts: list[str]) -> list[float]:
    pass

# Load the RAID test data
test_df = load_data(split='test')

# Run your detector on the dataset
predictions = run_detection(my_detector, test_df)

# Write predictions to a JSON file
with open('predictions.json') as f:
    json.dump(predictions, f)
```

Figure 9: A example showing how to use the RAID Pypi package to evaluate a detector on the dataset and submit it to the leaderboard.

is difficult for models.

Reddit Posts (Völske et al., 2017) This dataset contains reddit posts and their titles. We hypothesize that such data will be challenging to detect due to the first-person and informal style.

Movie Reviews (Maas et al., 2011) This dataset contains movie reviews from IMDb along with the names of the movies. The formality of the reviews are varied and this tests model's ability to recall details from movies as well as generate and detect opinionated text.

Wikipedia (Aaditya Bhat, 2023) This dataset contains introductions to various Wikipedia articles. This dataset is challenging as it tests the models ability to accurately recall facts relating to specific historical events.

Python Code (Raychev et al., 2016) This dataset contains python solutions to coding problems and the associated problem title. We include this as an initial foray into the detection of AI-generated code.

Czech News (Boháček et al., 2022) This domain consists of Czech language news articles. The topics and sources are diverse sampling from over 45 publications including mainstream journalistic websites, tabloids and independent news outlets in the Czech Republic.

German News (Schabus et al., 2017) This domain consists of German language news articles from DER STANDARD, an Austrian daily broadsheet newspaper. Articles are fairly political in nature covering topics such as the European migrant crisis, the 2016 Austrian presidential elections and the Syrian Civil War. We expect this dataset to test models' ability to generate opinionated political content in another language.

Paper Abstracts (Paul and Rakshit, 2021) This is a dataset of abstracts scraped from ArXiv together with paper titles. For this dataset and this dataset only, we filter the data such that only papers from 2023 or later are present in the data. This allows us to rule out the possibility that our models have memorized this text.

E.2 Generative Models

In Table 10 we list the exact generative models used in our project along with their unique identifiers. All open-source models were run using the HuggingFace `transformers` library (Wolf et al., 2020) and all closed-source models were run using

Dataset	Genre	Size
(Paul and Rakshit, 2021)	Abstracts	1966
(Bamman and Smith, 2013)	Books	1981
(Raychev et al., 2016)	Code	920
(Greene and Cunningham, 2006)	News	1980
(Arman, 2020)	Poetry	1971
(Bień et al., 2020)	Recipes	1972
(Völske et al., 2017)	Reddit	1979
(Maas et al., 2011)	Reviews	1143
(Aaditya Bhat, 2023)	Wiki	1979
(Boháček et al., 2022)	Czech	1965
(Schabus et al., 2017)	German	1970

Table 9: The number of articles sampled from each domain with their corresponding sources

Model	Identifier
GPT-2 (Radford et al., 2019)	`gpt2-xl`
MPT (+ *Chat*) (MosaicML, 2023)	`mpt-30b` `mpt-30b-chat`
Mistral (+ *Chat*) (Jiang et al., 2023)	`Mistral-7B-v0.1` `Mistral-7B-Instruct-v0.1`
LLaMA Chat (Touvron et al., 2023)	`Llama-2-70b-chat-hf`
Cohere (+ *Chat*) (Cohere, 2024)	`command (co.generate())` `command (co.chat())`
GPT-3 (Ouyang et al., 2022)	`text-davinci-002`
ChatGPT (OpenAI, 2022)	`gpt-3.5-turbo-0613`
GPT-4 (OpenAI, 2023)	`gpt-4-0613`

Table 10: The generative models used in our project

the proprietary APIs from Cohere[7] and OpenAI[8]. The following is a detailed list of the generative models used in the project.

GPT2 XL 1.5B (Radford et al., 2019) is a decoder-only model trained on the WebText dataset. This dataset consists of a collection of all documents that were linked from reddit posts or comments that had at least 3 or more upvotes. Released in February of 2019 and having 1.5B parameters, GPT2 is the predecessor of GPT3 and GPT4 and the most powerful open-source OpenAI model.

GPT3 (Ouyang et al., 2022) is a closed-source language model released by OpenAI on November 29th, 2022. The model was allegedly trained with a variety of data including the Common Crawl (filtered), WebText2, and Wikipedia datasets (Brown et al., 2020) but exact composition of the training dataset is unknown. It is the first model shown to work well with prompts and has shown great zero- and few-shot capabilities. In this study, we use the `text-davinci-002` model. We queried the model from November 1st to November 2nd 2023. Unfortunately, as of January 4th 2024, this model is no longer available for use on the OpenAI API. This is unfortunate as it prevents us from expanding the domains in future releases. We encourage researchers to keep this in mind when using OpenAI models for their research projects.

ChatGPT (OpenAI, 2022) is a version of GPT3 fine-tuned using Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022). We use the June 13th 2023 checkpoint of the model (`gpt-3.5-turbo-0613`). Although the number of parameters is unknown, ChatGPT demonstrates outstanding capability in language and code generation.

GPT4 (OpenAI, 2023) is the latest iteration of OpenAI's GPT family of models and is one of the largest and most powerful language models available to date. In this study, we use the `gpt-4-0613` checkpoint of the model through the `ChatCompletion` interface[9]. We queried the model from November 1st to November 2nd 2023.

LLaMA 2 70B (Touvron et al., 2023) is a decoder-only model trained by Meta (Facebook) and is the second model in the LLaMA series. Released on July 18th 2023, LLaMA 2 is the successor to the original LLaMA model which was trained on webpages from CommonCrawl, multilingual Wikipedia, books from Project Gutenberg, and QAs from Stack Exchange. The composition of LLaMA 2's training data is not known but it has shown impressive performance on many open-source evaluations and is widely considered competitive with the open-source state-of-the-art.

Mistral 7B (Jiang et al., 2023) is a decoder-only model trained by Mistral and is the first model released by the company. Released on September 27, 2023, Mistral 7B outperforms LLaMA 2 13B across various benchmarks at half the size. While model weights are open-source, the training data

[7] `https://docs.cohere.com/reference/about`
[8] `https://platform.openai.com/docs/introduction`
[9] `https://platform.openai.com/docs/guides/chat`

for the model is not and no details have been released regarding the makeup of this data.

MPT 30B (MosaicML, 2023) is a decoder-only model trained by Mosaic and is the first model released by the company. Released on June 22nd 2023, MPT 30B has an 8k context window and outperforms GPT-3 on various reasoning tasks. Training data consists of deduplicated C4 (Raffel et al., 2020; Lee et al., 2022), the RedPajama[10] split of CommonCrawl, and selected programming languages from The Stack (Kocetkov et al., 2022).

Cohere *command* (Cohere, 2024) is a closed-source model trained and released by Cohere. While original versions of this model were quoted as having roughly 50 billion parameters (Liang et al., 2023b), the size and training data of the current version of the Cohere model is unknown. Unlike OpenAI, Cohere does not version the *command* model and so, much like with `text-davinci-002` we are unable to expand our dataset to new domains without re-generating all generations. We queried the command model through both the `co.generate()` and `co.chat()` endpoints from November 1st to November 2nd 2023.

E.3 Prompts

In Table 12 we report the prompt templates for each of the 8 domains used in our project. We specifically avoided biasing the model towards a particular length or style of generation and did not use any of the text from the human-written documents other than the title when prompting.

To decide on the particular form of our prompts, we conducted two rounds of manual review. These rounds consisted of the authors determining problematic prompts by manually reviewing 10 generations per model in each domain for instances of degenerate repetition, meta-commentary or other signs of generated output. In problematic domains, we conducted individual explorations to determine if there existed some high level prompting concept that removed the unintended behavior and whether or not using such a prompt style caused regressions across other generators. After identifying a desired prompting change, we re-wrote our templates and restarted the review process.

Through this investigation, we found that continuation-style models benefit greatly from explicitly stating the source website in the prompt and that chat-style models benefit from language explicitly asking them not to repeat the title of the article (See Table 12). While we were unable to remove all instances of degenerate repetition or meta-commentary, this investigation significantly increased the quality of our dataset and we encourage future work on dataset creation to conduct a similar process when engineering their prompts.

[10] `https://www.together.ai/blog/redpajama-data-v2`

Attack	θ	Source
Alternative Spelling	100%	(Liang et al., 2023c)
Article Deletion	50%	(Liang et al., 2023a; Guerrero et al., 2022)
Homoglyph	100%	(Wolff, 2020; Gagiano et al., 2021)
Insert Paragraphs	50%	(Bhat and Parthasarathy, 2020)
Number Swap	50%	(Bhat and Parthasarathy, 2020)
Paraphrase	100%	(Krishna et al., 2023; Sadasivan et al., 2023)
Misspelling	20%	(Liang et al., 2023a; Gagiano et al., 2021; Gao et al., 2018)
Synonym	50%	(Pu et al., 2023a)
Upper Lower	5%	(Gagiano et al., 2021)
Whitespace	20%	(Cai and Cui, 2023; Gagiano et al., 2021)
Zero-Width Space	100%	(Guerrero et al., 2022)

Table 11: The adversarial attacks used in the project. θ represents the manually determined fraction of available attacks carried out. We determine this fraction through manual review.

E.4 Adversarial Attacks

In Table 11 we list the attacks used along with the attack rate θ and the relevant sources for the attacks. In this section we will list the attacks in more detail and discuss the various design decisions made. Implementations for all attacks can be found in our GitHub repository.

Alternative Spelling We use an American to British English dictionary[11] to construct a mapping between American and British spellings of words. We then find all instances of such words in the generation (defaulting to the longest available match if there were multiple substring matches for the same token). We then randomly sample a fixed percentage θ of the possible mutations to make with a set seed and apply the attack at those indices.

Article Deletion We search through the text and find every instance of the articles "a", "an", and

[11] `https://github.com/hyperreality/American-British-English-Translator`

	Continuation-Style Prompt	Chat-Style Prompt
Abstracts	The following is the full text of the abstract for a research paper titled "{title}" from arxiv.org:	Write the abstract for the academic paper titled "{title}".
Books	The following is the full text of a plot summary for a novel titled "{title}" from wikipedia.org:	Write the body of a plot summary for a novel titled "{title}". Do not give it a title.
Code	The following is the full text of a Python code solution to the exercise "{title}" from stackoverflow.com:	Write just the Python code solution for the problem "{title}".
German	Schreiben Sie einen Nachrichtenartikel mit dem Titel "{title}".	Es folgt ein Nachrichtenartikel mit dem Titel "{title}":
Czech	Následuje článek s názvem "{title}":	Napiš text článku, který má nadpis "{title}".
News	The following is the full text of a news article titled "{title}" from bbc.com:	Write the body of a BBC news article titled "{title}". Do not repeat the title.
Poetry	The following is the full text of a poem titled "{title}" from poemhunter.com:	Write the body of a poem titled "{title}". Do not repeat the title.
Recipes	The following is the full text of a recipe for a dish called "{title}" from allrecipes.com:	Write a recipe for "{title}".
Reddit	The following is the full text of a post titled "{title}" from reddit.com:	Write just the body of a Reddit post titled "{title}". Do not repeat the title.
Reviews	The following is the full text of a review for the movie "{title}" from IMDb.com:	Write the body of an IMDb review for the movie "{title}". Do not give it a title.
Wiki	The following is the full text of an article titled "{title}" from wikipedia.com:	Write the body of a Wikipedia article titled "{title}".

Table 12: The text of the generation prompts for all ten datasets in both continuation and chat style. The field {title} was replaced with the title of the book, dish, or news article before being passed into the generative model.

"the". We then randomly sample a fixed percentage θ of the possible mutations to make with a set seed and apply the attack at those indices.

Homoglyph Homoglyphs are character that are non-standard unicode characters that strongly resemble standard English letters. These are typically characters used in Cyrillic scripts. We use the set of homoglyphs from Wolff (2020) which includes substitutions for the following standard ASCII characters: a, A, B, e, E, c, p, K, O, P, M, H, T, X, C, y, o, x, I, i, N, and Z. We limit ourselves to only homoglyphs that are undetectable to the untrained human eye, thus we are able to use an attack rate of $\theta = 100\%$ and apply the attack on every possible character. For characters that have multiple possible homoglyphs we randomly choose between the homoglyphs.

Insert Paragraphs For this attack, we again split sentences using Punkt (Kiss and Strunk, 2006) and construct a list of all inter-sentence spans. We then sample θ percent of those spans and add the double newline character `\n\n` in-between the sentences to simulate a paragraph break.

Number Swap For this attack we use the following regular expression to extract all instances of numerical digits in the generation: `"\d+.?\d*"`. We then randomly select θ percent of these digits to modify and for each digit we randomly select an alternate number between 0 and 9 to replace the character with.

Paraphrase For paraphrasing we run the DIPPER-11B model[12] from Krishna et al. (2023) through HuggingFace (Wolf et al., 2020). DIPPER is a fine-tuned version of T5-11B (Raffel et al., 2020) specifically made for paraphrasing text to avoid machine-generated text detectors. We use the default settings from the paper, namely a sentence interval of 3 with lexical diversity of 60 and order diversity of 0. Since paraphrases are not inherently noticeable when models are correct, we are able to apply this attack across the entirety of the output text ($\theta = 100\%$).

Misspelling For this attack, we manually constructed a dictionary of common misspellings[13] and only applied the attack to instances of words that have misspellings in our dictionary. We did this to minimize suspicion from human readers, as

[12] https://huggingface.co/kalpeshk2011/dipper-paraphraser-xxl

[13] https://en.wikipedia.org/wiki/Commonly_misspelled_English_words

certain common words such as 'the' or 'him' are rarely misspelled. Instead of randomly selecting θ percent of the possible candidate words to misspell, we follow Gagiano et al. (2021) and misspell only the top θ percent most likely candidate words by log likelihood as determined by GPT 2 small. This allows us to choose only the most effective misspellings to apply.

Synonym For this attack we originally planned to use the DFTFooler algorithm from Pu et al. (2023a). However, we found the candidate synonyms to be of low quality and easily detectable by our manual analysis. Thus we decided to implement our own algorithm based largely on DFTFooler. Our algorithm produces high-quality and diverse synonym substitutions without relying on any of the large decoder-only language models used for generation.

We start by iterating over all tokens in the generation. For each token i we replace it with a mask token and get the top 20 most likely mask-fill candidates[14] according to BERT (Devlin et al., 2019). We then compute the part-of-speech tag for each candidate using NLTK (Bird and Loper, 2004) and reject all candidates that do not match the part-of-speech of the original token. We then get the static FastText (Bojanowski et al., 2017) embeddings for all candidate substitutions and reject all tokens that have cosine similarity of less than 0.5 with the original token. Doing this for each index i gives us a global list of all valid candidate swaps across the entire generated passage. From this list we select the $\theta \cdot L$ most likely synonym swaps according to BERT where L is the length (in tokens) of the passage.

The full code for this algorithm can be found in our project repository along with the implementations of the other adversarial attacks.

Upper-Lower This attack randomly selects some θ percent of the tokens in the passage and swaps the first letter of the token to be uppercase if it was lowercase and lowercase if it was originally uppercase.

Whitespace This attack randomly selects some θ percent of inter-token spaces and adds an extra space character inbetween the tokens. This can occasionally result in multiple spaces added between two tokens as sampling is done with replacement.

[14] In order to reduce computation time we limit BERT to a window of 20 tokens on either side of the index when determining the mask-fill candidates. We found no reduction in candidate quality from this modification.

Zero-Width Space The unicode zero-width space `U+200B` is a character that exists in text encoding but is not visible to human readers in most scenarios. Thus, for this attack we insert this character at every possible opportunity (before and after each visible character in the generation).

E.5 Repetition Penalty vs. Frequency Penalty vs. Presence Penalty

Given a temperature $T > 0$ and a set of scores $x_i \in \mathbb{R}^d$ for each token i in a vocabulary, the probability p_i of predicting the ith token is given by:

$$p_i = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)}$$

The repetition penalty defined by Keskar et al. (2019) modifies this distribution as follows:

$$p_i = \frac{\exp(x_i/(T \cdot I(i \in g)))}{\sum_j \exp(x_j/(T \cdot I(j \in g)))}$$

Where g is a list of previously generated tokens and $I(c) = \theta$ if c is True else 1. OpenAI implements a form[15] of this penalty (referred to as a 'presence penalty') which is additive instead of multiplicative:

$$p_i = \frac{\exp((x_i/T) - I(i \in g))}{\sum_j \exp((x_j/T) - I(j \in g))}$$

Cohere (as of May 13th 2024) provides no documentation on the nature of their presence penalty despite requests from the authors for the proper documentation.

E.6 Hardware

We ran our generations over the course of 15 days from November 1st 2023 to November 15th 2023 on 32 NVIDIA 48GB A6000 GPUs. We ran all models with 16-bit precision as we found that outputs were identical to full precision and it cut our inference time in half. The amount of GPU hours used by each family of models is as follows: LLaMA 2 70B (+ Chat) 8,376 hours, MPT (+ Chat) 5,440 hours, Mistral (+ Chat) 672 hours, GPT2 (+ Chat) 352 hours. In total we used 14,872 GPU hours (620 GPU days) to generate the RAID dataset.

[15] https://platform.openai.com/docs/guides/text-generation/parameter-details

F Detector Details

F.1 Detectors

In this section we provide a detailed description of all detectors used in the evaluations of the RAID dataset.

RoBERTa (GPT2) (Solaiman et al., 2019) This detector[16] is a RoBERTa model (Liu et al., 2019) fine-tuned on the GPT2 output dataset. This dataset consists of outputs from GPT2 in open domain settings with three different decoding strategies: greedy decoding, top-k=50, and fully random sampling and has been a baseline inclusion for many years. We use both the base and large size of this model in our comparisons.

RoBERTa (ChatGPT) (Guo et al., 2023) This detector is a RoBERTa-base model (Liu et al., 2019) fine-tuned on the HC3 dataset. HC3 consists of roughly 27,000 questions paired with both human and ChatGPT answers in various domains such as reddit, medicine, finance, and law. We download and query the detector via HuggingFace datasets with the unique identifier `Hello-SimpleAI/chatgpt-detector-roberta`

RADAR (Hu et al., 2023) This detector is a fine-tuned version of Vicuna 7B (which itself is a fine-tune of LLaMA 7B). It was trained in a generative adversarial setting alongside a paraphrase model. The paraphraser was trained specifically to fool the detector and the detector was trained to accurately detect generations from the paraphraser, human-text from the WebText dataset, and outputs from the original language model. We download and query this detector from HuggingFace with the unique identifier `TrustSafeAI/RADAR-Vicuna-7B`

GLTR (Gehrmann et al., 2019) Originally intended as an interface to help humans better detect generated text, GLTR has become a standard baseline in robustness studies of detector abilities. GLTR evaluates the likelihood of text according to a language model and bins tokens according to their likelihoods and uses these bins as features for detection. We use the default settings from the GLTR repository[17] namely our cutoff set at rank=10 and the language model set to GPT2 small.

FastDetectGPT (Bao et al., 2023) This detector is an improvement on the original DetectGPT (Mitchell et al., 2023)—speeding up inference by 340x without any reduction in accuracy. For the scoring model we use the repository default of GPT-Neo-2.7B and for the reference model we again use the default of GPT-J-7B. Since neither of these models were used to generate continuations in our dataset, we felt that this was a reasonable choice.

Binoculars (Hans et al., 2024) This detector uses perplexity divided by cross-entropy between two very similar language models as the metric for detection. In our implementation we use the code from the official GitHub repository and calculate perplexity using the default models from the repository, namely Falcon 7B and Falcon 7B Instruct (Almazrouei et al., 2023). Much like FastDetectGPT, since neither of these models were used to generate continuations, we believed this made for a fair comparison.

LLMDet (Wu et al., 2023) This detector computes the proxy-perplexity of the input text from 10 different small language models and uses these features for detection. The proxy-perplexity is an approximation of the true perplexity calculated by repeatedly sampling n-grams from models rather than by actually running them. Of the models used, none of them were used for generations in our dataset, thus this was a fair comparison.

GPTZero (Tian and Cui, 2023) GPTZero is a closed-source commercial detector released in January 2023 and was among the first to gain widespread media attention for their detection-as-a-service business model. We queried the detector on January 24th 2024 using the v2 API[18] and threshold on the `completely_generated_prob` field.

Originality Originality is a closed-source commercial detector released in November 2022 and was the first company to adopt the detection-as-a-service business model. We queried this detector from January 24th to 25th through the v1 API[19] and threshold on the 'score' field in the output JSON. For the Czech and German news domains, we specifically query the multilingual "version 3" of the detector.

Winston This detector is the closed-source commercial detector that claims the highest accuracy

[16] We download the model hosted by OpenAI from the following link https://openaipublic.azureedge.net/gpt-2/detector-models/v1/detector-large.pt

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

[18] https://api.gptzero.me/v2/predict/text

[19] https://api.originality.ai/api/v1/scan/ai

out of any detector (99.98%). We query this model through the v1 API[20] and specifically set the input language to German when we detect that our input text is in German. Unfortunately, since winston does not support detection in Czech, we use English as the input language for the Czech news domain. We once again threshold on the 'score' field in the output JSON.

ZeroGPT This detector is the final commercial detector. We run this as it is the only detector besides GPTZero that has been evaluated in another benchmark paper. We query the API at the following link[21] and use the 'isHuman' return field as the classifier output.

F.2 Thresholds

In order to find the thresholds that achieve a fixed false positive rates we had to implement some basic search procedure. We searched our thresholds in a linear fashion: We start at the threshold corresponding to the mean score of human data (50% FPR) and approach the desired false positive rate by iteratively incrementing or decrementing the threshold. If we overshoot the target fpr we divide our step size in half and flip the sign. We continue to do this until the false positive rate is within $\epsilon = 0.0005$ of the desired false positive rate or until 50 iterations are reached. If 50 iterations are reached without convergence then we select the threshold corresponding to the FPR that is closest to the target while still being less than the target. If there is no such threshold then we note that the detector could not reach the target FPR and simply choose the value closest to and greater than the target. In Table 13 we list the thresholds our algorithm found for each detector along with the exact false positive rates that these thresholds allow.

G Extended Figures and Tables

G.1 Dataset Statistics and Evaluations

In Figures 10 and 11 we report extended statistics about the dataset. We see some interesting trends here, namely that human-written text is still significantly less likely than machine-generated text according to LLaMA 2 7B and that it is also the least repetitive. These results push back on claims that language models are approaching human-level performance and therefore detection is unreasonably difficult.

[20] https://api.gowinston.ai/functions/v1/predict

[21] https://api.zerogpt.com/api/detect/detectText

G.2 Extended Heatmaps

In Figure 12 we show the extended heatmaps from Figure 6 in the main paper. We see that the trend holds that RoBERTa GPT2 is significantly better on GPT2 generations and that RADAR is uncharacteristically bad on IMDb Movie Reviews.

G.3 Model Performance vs. Domain

In Table 14 we report the results of our 12 detectors across all different domains of generated text. We see that the metric-based methods such as Binoculars and FastDetectGPT generalize surprisingly well across domains. We also see that in general detectors perform well but occasionally perform surprisingly poorly.

G.4 Detector Accuracy vs. Decoding Strategy

In Table 5 we report the results of our evaluation broken up by category of model and by decoding strategy. Much like in Figure 5 from the main paper, we see that certain combinations of decoding strategies and models can cause detector accuracy to plummet unexpectedly. This raises serious concerns for the robust deployment of detectors.

G.5 Detector Accuracy vs. Adversarial Attack

In Table 16 we report the full version of the results from Table 6 in the main paper. We see similar trends, namely that certain adversarial attacks work better on certain detectors and that occasionally adversarial attacks actually improve detector performance rather than harm it. One notable inclusion here is the zero-width space attack, which seems to either cause detectors to assign all positive or all negative labels. Future work should investigate this phenomenon.

H Example Generations

In Table 17 and 18 we provide example outputs for each generative model and adversarial attack. We see that different generative models have significantly differing styles, underscoring the difficulty of the detection problem.

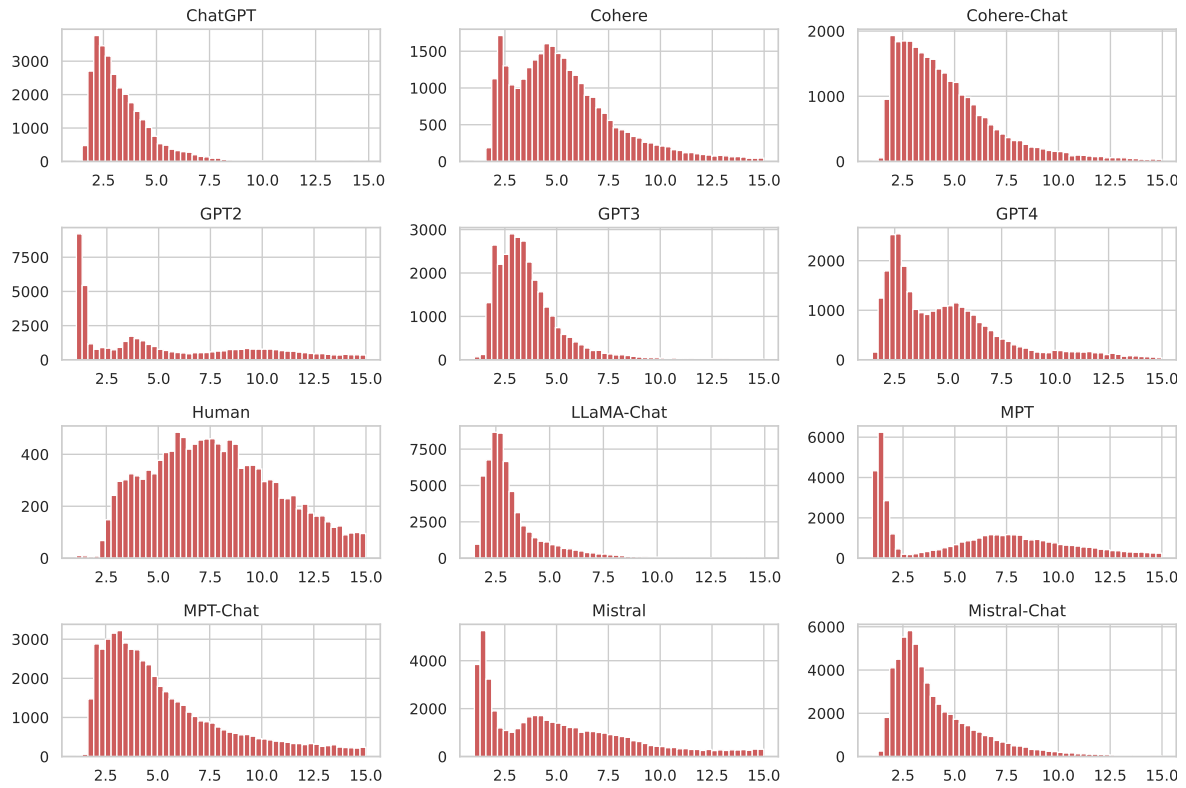

Figure 10: Histograms of perplexity according to LLaMA 2 7B per generative model in the dataset. We see that there is still a significant difference between human-written and machine-generated text with respect to perplexity.

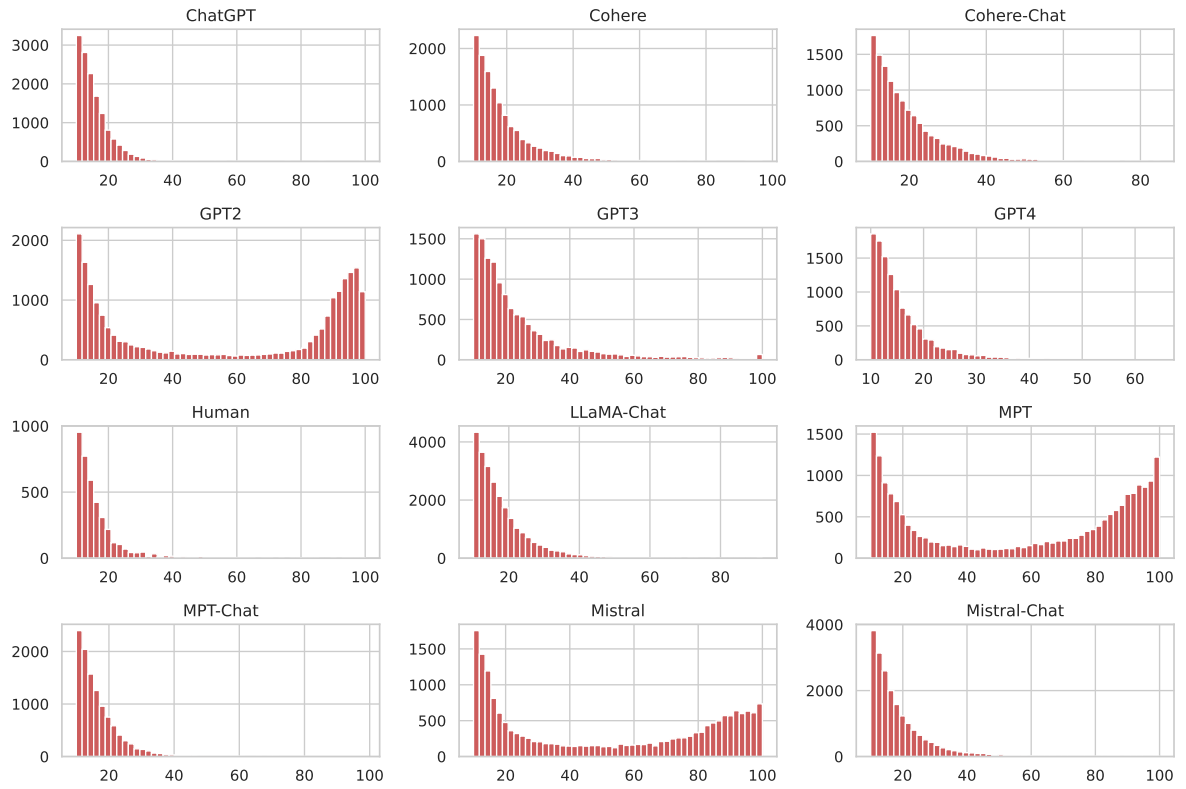

Figure 11: Histograms of SelfBLEU (Zhu et al., 2018) per generative model in the dataset. We see that continuation models tend to be more repetitive than chat models and that human-written text is by far the least repetitive.

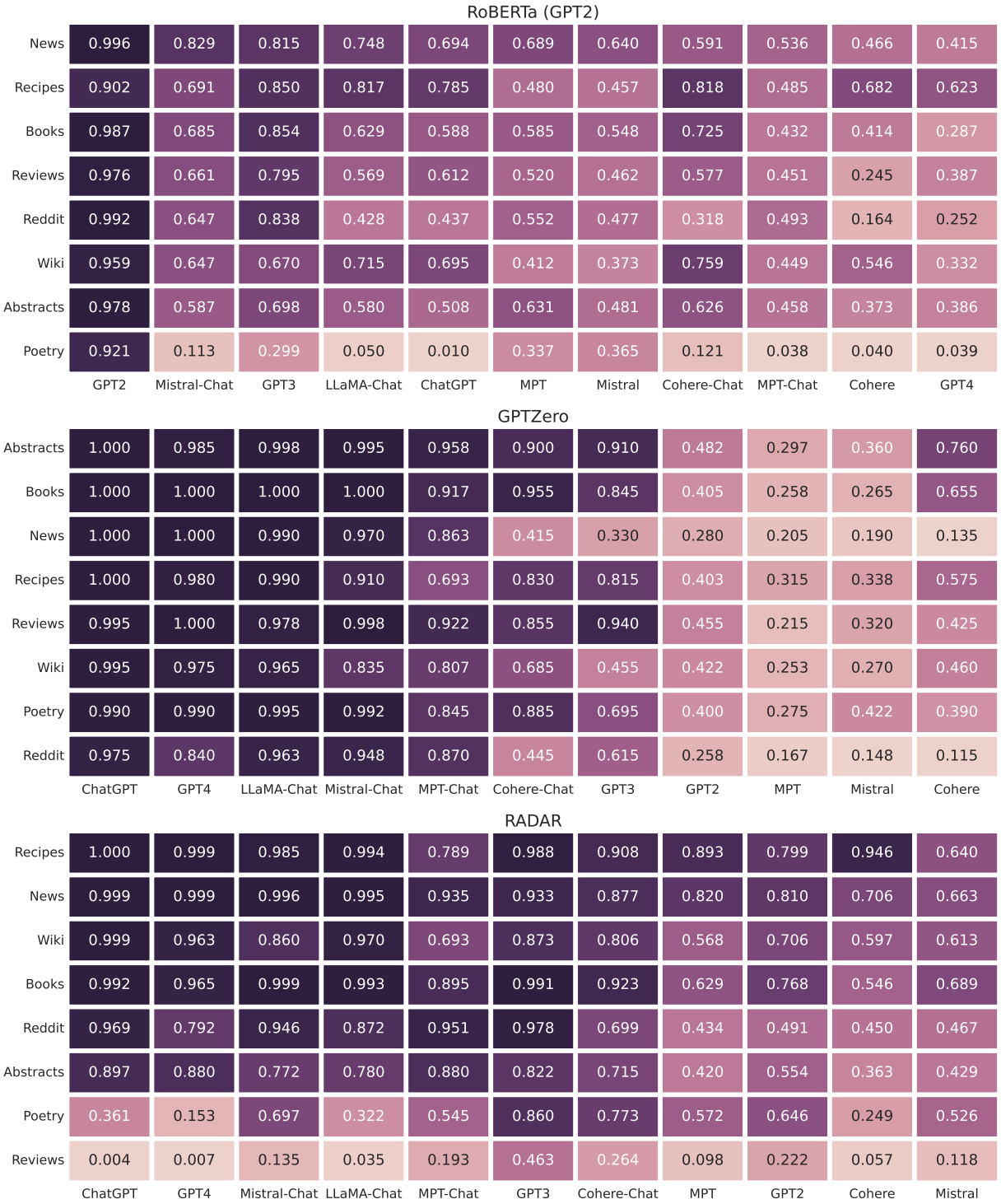

Figure 12: Extended heatmap of RoBERTa GPT2, GPTZero, and RADAR's performance across all models and domains in the RAID dataset. We see that the trends noted in Figure 6 still hold.

	News	Wiki	Reddit	Books	Abstracts	Reviews	Poetry	Recipes
RoBERTa-B GPT2	0.032 (5.0%)	0.379 (5.0%)	0.477 (5.0%)	0.586 (5.0%)	0.055 (5.0%)	0.539 (5.0%)	0.998 (5.0%)	0.916 (5.0%)
RoBERTa-L GPT2	0.070 (5.0%)	0.459 (5.0%)	0.100 (5.0%)	0.161 (5.0%)	0.085 (5.0%)	0.298 (5.1%)	0.762 (5.0%)	0.315 (5.0%)
RoBERTa-B C-GPT	0.987 (5.0%)	0.983 (5.0%)	0.219 (5.0%)	0.996 (5.0%)	0.007 (5.0%)	0.371 (5.0%)	0.295 (5.0%)	0.998 (5.0%)
RADAR	0.022 (5.0%)	0.061 (5.0%)	0.695 (5.0%)	0.174 (5.0%)	0.31 (5.0%)	0.997 (5.0%)	0.457 (5.0%)	0.016 (5.0%)
GLTR	0.788 (5.0%)	0.788 (5.0%)	0.767 (5.0%)	0.742 (5.0%)	0.726 (5.0%)	0.757 (5.0%)	0.756 (5.0%)	0.863 (5.0%)
FastDetectGPT	0.920 (4.8%)	0.870 (5.1%)	0.870 (5.1%)	0.930 (4.9%)	0.860 (4.6%)	0.900 (4.8%)	0.940 (5.9%)	0.880 (5.5%)
LLMDet	1.000 (5.0%)	1.000 (5.0%)	1.000 (5.0%)	1.000 (5.0%)	0.999 (5.0%)	1.000 (5.1%)	1.000 (5.0%)	0.998 (5.0%)
Binoculars	0.077 (4.9%)	0.093 (5.0%)	0.099 (5.0%)	0.085 (5.0%)	0.092 (5.0%)	0.097 (4.9%)	0.084 (5.0%)	0.094 (5.0%)
GPTZero	0.047 (5.0%)	0.032 (5.0%)	0.057 (5.0%)	0.125 (5.0%)	0.125 (5.0%)	0.070 (5.0%)	0.031 (5.0%)	0.035 (5.0%)
Originality	0.375 (5.0%)	0.938 (5.0%)	0.250 (5.0%)	0.312 (5.0%)	0.257 (5.0%)	0.461 (5.0%)	0.047 (5.0%)	0.750 (5.0%)
Winston	0.001 (4.0%)	0.970 (5.0%)	0.062 (5.0%)	0.998 (5.0%)	0.000 (6.0%)	0.062 (5.0%)	0.875 (5.0%)	0.996 (5.0%)
ZeroGPT(*)	1.000 (29.0%)	1.000 (48.0%)	0.375 (1.0%)	0.250 (9.0%)	0.500 (4.0%)	1.000 (5.0%)	0.125 (5.0%)	1.000 (52.0%)

Table 13: Thresholds found by our search and the exact False Positive Rates on our dataset. We see that ZeroGPT is incapable of achieving the target FPR of 5% in many domains.

	News	Wiki	Reddit	Books	Abstracts	Reviews	Poetry	Recipes
RoBERTa-B GPT2	74.3	64.7	56.3	67.9	56.3	69.1	23.9	64.6
RoBERTa-L GPT2	69.8	59.5	54.1	62.4	58.9	58.2	24.4	67.2
RoBERTa-B CGPT	45.1	48.7	44.6	53.5	72.3	65.3	32.0	5.9
RADAR	88.0	76.8	71.8	84.5	66.7	14.1	53.0	**88.5**
GLTR	66.9	64.3	65.7	74.0	62.0	67.3	34.8	67.2
FastDetectGPT	74.2	77.3	70.9	76.2	76.4	77.5	63.4	74.6
LLMDet	39.8	32.6	39.6	37.1	18.0	33.1	30.7	48.1
Binoculars	80.7	76.7	79.4	83.7	79.1	80.1	**81.0**	76.6
GPTZero	58.1	62.8	57.0	71.4	74.9	70.5	69.5	67.6
Originality	**88.4**	**83.2**	**85.0**	**90.4**	87.7	**87.3**	75.1	82.8
Winston	72.4	54.9	68.9	70.7	**94.7**	72.9	64.3	68.9
ZeroGPT(*)	72.2	70.6	65.1	73.3	60.3	68.6	50.0	63.7

Table 14: Accuracy Score at FPR=5% for detectors across different domains. We see that metric-based methods perform surprisingly well across domains and that detectors can perform surprisingly poorly on unseen domains.

	GPT2	GPT3	ChatGPT	GPT4	Cohere		Mistral		MPT		Llama	Total
Chat? (Y/N)	✗	✗	✓	✓	✗	✓	✗	✓	✗	✓	✓	-
R-B GPT2	84.0	74.7	65.4	42.4	42.9	61.1	45.7	65.9	49.2	45.9	68.7	59.1
R-L GPT2	96.3	72.4	53.7	33.8	37.3	56.6	47.6	60.5	52.6	41.6	56.7	56.7
R-B CGPT	36.7	54.2	66.1	30.0	31.3	49.6	17.7	69.9	17.8	53.3	70.1	44.8
RADAR	64.7	88.6	82.1	76.0	51.4	77.2	54.1	83.6	58.0	76.5	78.5	70.9
GLTR	66.6	85.1	81.4	53.7	54.2	67.4	49.1	75.4	35.4	52.5	81.6	62.6
F-DetectGPT	72.1	95.4	96.1	73.9	84.7	85.1	58.2	81.3	45.3	57.1	94.0	73.6
LLMDet	48.2	40.2	18.9	27.0	32.6	35.6	31.5	35.7	28.2	21.4	55.1	35.0
Binoculars	68.9	**99.2**	**99.6**	91.9	**94.8**	**95.4**	62.3	91.7	45.2	70.8	97.6	79.6
GPTZero	38.8	70.1	99.4	97.1	43.9	74.6	28.9	**95.6**	24.8	85.9	**98.5**	66.5
Originality	**99.1**	98.2	98.2	89.9	78.9	90.6	**71.0**	95.5	**58.1**	76.2	94.7	**85.0**
Winston	47.6	77.8	**99.6**	**98.8**	63.6	86.2	46.1	94.9	24.8	**79.2**	97.5	71.0
ZeroGPT(*)	42.4	90.2	93.2	67.1	65.9	76.6	49.3	81.4	27.3	66.0	93.7	65.5

Table 15: Accuracy at FPR=5% for detectors on non-adversarial outputs of different models. We see that base models are more difficult to detect than their chat fine-tuned counterparts and that metric-based methods show impressive cross-model generalization. Asterisks (*) indicate that the detector was unable to achieve the target FPR.

	None	AS	AD	HG	IP	NS	PP	MS	SYN	ULS	WSA	ZWS
RoB-B GPT2	59.1	55.6	37.1	7.6	56.9	55.9	68.9	43.8	71.5	18.8	45.2	99.9
RoB-L GPT2	56.7	52.4	33.2	21.3	55.1	51.7	72.9	39.5	79.4	19.3	40.1	99.9
RoB-B CGPT	44.8	43.3	38.0	0.0	5.2	44.3	49.2	42.1	39.6	31.7	0.1	0.0
RADAR	70.9	70.8	67.9	59.3	73.7	71.0	67.3	69.5	67.5	70.4	66.1	82.2
GLTR	62.6	61.2	52.1	24.3	61.4	59.9	47.2	59.8	31.2	48.1	45.8	97.2
F-DGPT	73.6	71.6	64.7	51.4	72.0	68.2	71.8	70.7	34.0	60.4	64.4	98.9
LLMDet	35.0	33.9	27.4	40.6	27.2	33.8	28.5	32.7	27.3	23.4	4.4	27.1
Binoculars	79.6	78.2	74.3	37.7	71.7	77.1	80.3	78.0	43.5	73.8	70.1	99.1
GPTZero	66.5	64.9	61.0	66.2	66.2	65.8	64.0	65.1	61.0	56.5	66.2	66.2
ZeroGPT	65.5	65.4	59.7	82.4	64.9	64.7	46.7	64.7	18.8	54.5	64.2	48.0
Originality	85.0	83.6	71.4	9.3	85.1	86.0	96.7	78.6	96.5	75.8	84.9	4.9
Winston	71.0	68.9	66.9	26.3	69.8	69.0	52.6	67.5	63.6	56.8	46.8	25.0

Table 16: Accuracy Score at FPR=5% for all detectors across different adversarial attacks. Abbrevations are: AS: Alternative Spelling, AD: Article Deletion, HG: Homoglyph, IP: Insert Paragraphs, NS: Number Swap, PP: Paraphrase, MS: Misspelling, SYN: Synonym Swap, ULS: Upper Lower Swap, WSA: Whitespace Addition, ZWS: Zero-Width Space Addition

Domain: Reddit, **Title**: VW album track list/lyric sheet inquiry? (**ID**: 25fb9f14-c7c5-4e76-95ac-bf149780f30d)	
Prompts: (**Chat**) Write just the body of a Reddit post titled "VW album track list/lyric sheet inquiry?". (**Non-Chat**): The following is the full text of a post titled "VW album track list/lyric sheet inquiry?" from reddit.com	
Original Reddit Post (Human)	I bought a "Modern vampires of the city" on vinyl recently, and I was wondering if anyone had information on the nature of the track list? Its on the backside of the poster, and its a normal tracklist and lyric sheet except there's some things circled and underlined and stuff. I looked it up, and I thought origionally that it was supposed to look like a playbook, like football, but it looks like the marks are intentional to the numbers and stuff so then I thought, maybe it was them planning their album? And they just decided to add some flair to the track list? Either way, im curious, and a quick Google search didn't work, so I figured you guys would know better than me. I would love any input I could get
GPT-2	"Hi, this is a thread for people of any ability to help me with a query related to the following track: "Cherries On The Wind" from the album 'Volkswagen', which was released July 2008 to August 2009. I have a few questions about all of them (my two biggest questions are 'Who sings it?' and 'Who wrote it and why?'): (1) Is the singer named "Cherries" a female? If not, is it a boy, female or something else? (2) Is it a female lead or male supporting singer in the band? (3) Who writes the lyrics which are written in both German and English, in the style of a pop song? Is it a German name or is it the name of a song artist, musician or songwriter? (4) Is there a female vocalist in the band? If so, can you shed some light on the singer? Are there German solo projects by her? If so, can you shed some light on the singer? Are there German solo projects by her? http://www.youtube.com/watch?v=jY9FyMwKvCk&feature=fvrs If you need to ask more question, just add them to the top of the post and I'll answer. Thanks much! 2 people have voted this message helpful Lyrics: https://www.youtube.com/watch?v=fSdWJ4Wy7k0
GPT-3	Hey all, I'm wondering if anyone might be able to help me out with something. I'm trying to track down a copy of the lyric sheet/track list for the album "Volkswagen." I've been able to find a few images online, but they're all partial and/or low quality. If anyone has a high quality scan or knows where I can find one, I would really appreciate it! Thanks in advance!
ChatGPT	Hello everyone, I recently got my hands on a vinyl copy of VW's latest album and I am absolutely loving it! The production, the melodies, and the lyrics are just top-notch. I've been obsessively listening to it and can't get enough. However, I was wondering if anyone has access to or knows where I can find the track list and lyric sheet for the album? I've been searching online but haven't had much luck so far. I would really love to dive deeper into the lyrics and understand the meaning behind some of the songs. If anyone can point me in the right direction or has any information, I would greatly appreciate it. Thanks in advance!
GPT-4	Hello fellow VW fans! I've recently got my hands on a vintage vinyl copy of one of the older Vampire Weekend albums. However, it didn't come with a track list or lyric sheet. It's proving to be a bit of a task to identify some of the tracks and I'd love to sing along to my favourites. If anyone could point me to where I might find a track list and/or a place where I can find the lyrics to the songs, I would greatly appreciate it. Let me know or feel free to message me directly. Thanks for your help, guys!
Mistral	> This has been bugging me for a bit. I never got my copy of "VW" (or "VW EP" - whichever you want to call it) so I never had a chance to look at the lyric sheet. I know there's an extended version on iTunes, but I haven't the funds to purchase it since the EP and "TNWT" came out at the same time. But I would like to know the lyrics to at least "Wonderful Life" (for which I know there's no lyrics of), "The New Wave", "Song of Our Hits", "What If God Was One Of Us", and "The Hits Just Keep Coming". I'd like to know how the song "What If God Was One Of Us" fits with the storyline (or is it related at all?), as well as the whole thing behind "What Would She Say?", and the ending song that has to be either a spoiler for "VW" ("I Am the Wind"), or something relating to "TNWT" (I think that line from the "TNWT" lyric sheet about a "new power" in the verse between "The Hits Just Keep Coming" and "Song For Our Hits" is what I'm thinking of)
Mistral Chat	Hey everyone, I just got into The Velvet Underground and I've been a huge fan of their work ever since. I've been doing a deep dive into their discography and I was wondering if anyone had a track list or lyrics sheet for the album "VW"? I can't find it anywhere online and it's driving me crazy. Any help would be greatly appreciated! Thanks in advance!
Cohere	VW album track list/lyric sheet inquiry? Hello! I was wondering if anyone had scanned/digitized the track list or lyric sheet from the VW album (oil diffuser). I know it's floating around here somewhere, but damned if I can find it. Thanks!
Cohere Chat	Check this out! Been waiting for this for ages!! Hey all – I've recently fallen in love with VW's latest album (thank you Reddit for the recommendation!) but I cannot for the life of me find the track list or lyrics anywhere. The songs are amazing but still quite new for me, so I have no clue what the lyrics are and wanna know!!! Does anyone have a track list/ lyric sheet that they could share? Any help is appreciated!

MPT	I read this on Reddit and think it bears repeating and expanding upon on a site that VW followers frequent, and I figured it behooved me to get it out to you guys ASAP The first batch of test pressings for the new Volcano Witch album, which was recorded during a recent trip to Chicago by VW member John McEntire, will be arriving very soon. Before this happens, the band wanted to know whether or not our fans will be interested in a test pressing lyric sheet. This is a special edition (limited number) item that will only be made available to those supporters that were very instrumental to the creation & completion of the new VW album. The test pressing lyric sheet will feature all 10 tracks with full handwritten lyrics by John, who also took the time to hand write lyrics to two new never heard before VW songs, which will only be available on the lyric sheet & test pressings of the album. If our followers are interested in this special edition item please submit a screenshot of the below image with the word “YES” placed in the comment section of this post. If you’re unable to submit a screenshot simply place the word “YES” in the comment section of this post & I’ll make sure you’re added to list for the special test pressing lyric sheet. All proceeds from the test pressing will be donated to The Center For Discovery, the school my daughter attends, in her honor... In the meantime, while the band waits to hear from their fans, we’re giving you the chance to be one of 100 lucky winners to win a signed “Storm Lord” poster that was created by the band’s longtime collaborator Eric Adrian Lee, who also designed & made all the artwork for the new Volcano Witch album. To enter yourself into this giveaway all you have to do is share this post. That’s it, simple as that. We’ll pick the lucky winners on October 2nd & then post the giveaway winners on our social media. So... are you interested in the special test pressing lyric sheet? Source: http://redd.it/30fsl1 Posted at 10:47 PM in Volcano Witch | Permalink | Comments (0) Volcano Witch - "A.M. (Live at the Empty Bottle)"
MPT Chat	Hey everyone! I’m a big fan of VW and I was wondering if anyone knows where I can find the album track list and lyric sheet for their latest album? I’ve been searching online but can’t seem to find it anywhere. Any help would be much appreciated! Thanks in advance!
Llama Chat	Hey fellow Redditors, I’m hoping someone can help me out with a little inquiry I have about the Velvet Underground’s album "The Velvet Underground & Nico". I’ve been trying to find a track list/lyric sheet for the album that includes the bonus tracks from the CD reissue, but I can’t seem to find one anywhere. I’ve searched high and low on Google, but all the track lists I come across only include the original track list for the album, without the bonus tracks. I’ve also checked the album’s Wikipedia page and various other websites, but no luck. Does anyone know of a resource or website that has the complete track list/lyric sheet for the album, including the bonus tracks? I’d really appreciate it if someone could point me in the right direction. Thanks in advance for the help, and sorry if this has been asked before (I searched the subreddit but didn’t find anything).

Table 17: Output generations from each model in RAID for the Reddit post titled “VW album track list/lyric sheet inquiry?” using random sampling (temp=1, p=1) and no repetition penalty. Prompts used are listed in Table 12

Domain: Reviews, **Model**: Cohere, **Decoding**: Sampling (No Penalty), (**ID**: b0aa73c4-ff31-4a43-9472-8fe7a85e2754) **Prompt**: The following is the full text of a review for the movie "Fast Five" from IMDb.com:	
Original (No Attack)	This is probably my favorite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a great idea, making this franchise better than ever. The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Alternative Spelling	This is probably my favourite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a great idea, making this franchise better than ever. The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Article Deletion	This is probably my favorite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was ~~a~~ great idea, making this franchise better than ever. The action was great and suspenseful, and ~~the~~ new cast members were amazing. This is a must-see.
Homoglyph	This is probably my favorite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a great idea, making this franchise better than ever. The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Insert Paragraphs	This is probably my favorite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a great idea, making this franchise better than ever. [¶] The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Number Swap	This is probably my favorite action movie of 0346, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a great idea, making this franchise better than ever. The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Paraphrase	This is surely the best of all the action films in 2011. I'm really looking forward to Fast & Furious 6. I loved that they brought back Vin-Diesel and the late Paul-Walker, it made the series better than ever. The action was suspenseful and exciting and the new actors were all great. This is a must-see.
Misspelling	This is probably my favorite action movie of 2011, can't wait to see Fast and Furious 6. Bringing back Vin Diesel and the late Paul Walker was a grat idea, making this franchise better than ever. The action was grate and suspenseful, and the new cast members were amazing. This is a must-see.
Synonym Swap	This is also my favourite action film of 2011, can't wait to see Fast and Furious 6. Taking down Vin Diesel and the late Scott Davis was a good thing, doing this franchise better than ever. The action was good and suspenseful, and the new cast members were amazing. This is a must-see.
Upper Lower	This is probably my favorite action movie of 2011, can't wait to see Fast and furious 6. Bringing back Vin Diesel and the late Paul walker was a great idea, making this franchise better than ever. The action was great and suspenseful, and the new cast members were amazing. This is a must-see.
Whitespace	This is probably my favorite[ ] action movie of 2011, can't wait to see Fast and[ ] Furious 6. Bringing[ ] back Vin Diesel[ ] and the late Paul Walker was a great idea,[ ] making this[ ] franchise better[ ] than ever. The action was great and suspenseful, and the new cast members[ ] were amazing. This is a must-see.
Zero-Width Space	T[U+200B]h[U+200B]i[U+200B]s[U+200B] i[U+200B]s[U+200B] [U+200B]p[U+200B]r[U+200B]o[U+200B]b[U+200B]a[U+200B]b[U+200B]l[U+200B]y[U+200B] [U+200B]m[U+200B]y[U+200B] [U+200B]f[U+200B]a[U+200B]v[U+200B]o[U+200B]r[U+200B]i[U+200B]t[U+200B]e[U+200B] [U+200B]a[U+200B]c[U+200B]t[U+200B]i[U+200B]o[U+200B]n[U+200B] [U+200B]<...>

Table 18: Example outputs for each adversarial attack in RAID when applied to the IMDb movie review for "Fast Five" generated by Cohere using random sampling (temp=1, p=1) and no repetition penalty. Blue color indicates the portion of the text that was changed by the attack. Detailed descriptions of each attack along with their effective attack surfaces are listed in Appendix E.4.