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).

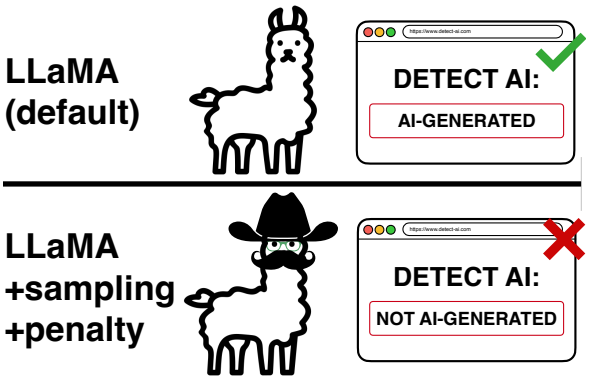

Figure 1: Detectors for machine-generated text are often highly performant on default model settings but fail to detect more unusual settings such as using random sampling with a repetition penalty.

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

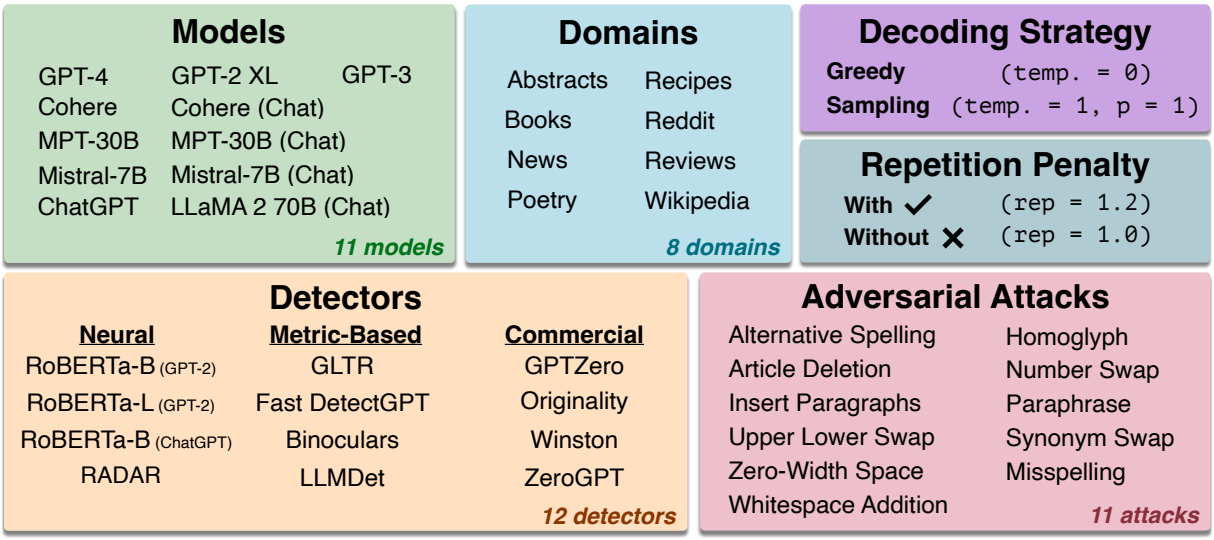

Figure 2: An overview of the structure of the RAID dataset. We generate 2,000 continuations for every combination of domain, model, decoding, penalty, and adversarial attack. This results in roughly 6.2 million generations for testing. We then evaluate each detector on all pieces of generated text in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