

Red Teaming Language Model Detectors with Language Models

Zhouxing Shi*, Yihan Wang*, Fan Yin*, Xiangning Chen, Kai-Wei Chang, Cho-Jui Hsieh

University of California, Los Angeles

{zshi, yihanwang, fanyin20, xiangning, kwchang, chohsieh}@cs.ucla.edu

Abstract

The prevalence and strong capability of large language models (LLMs) present significant safety and ethical risks if exploited by malicious users. To prevent the potentially deceptive usage of LLMs, recent work has proposed algorithms to detect LLM-generated text and protect LLMs. In this paper, we investigate the robustness and reliability of these LLM detectors under adversarial attacks. We study two types of attack strategies: 1) replacing certain words in an LLM’s output with their synonyms given the context; 2) automatically searching for an instructional prompt to alter the writing style of the generation. In both strategies, we leverage an auxiliary LLM to generate the word replacements or the instructional prompt. Different from previous works, we consider a challenging setting where the auxiliary LLM can also be protected by a detector. Experiments reveal that our attacks effectively compromise the performance of all detectors in the study with plausible generations, underscoring the urgent need to improve the robustness of LLM-generated text detection systems. Code is available at <https://github.com/shizhouxing/LLM-Detector-Robustness>.

1 Introduction

Large language models (LLMs), such as ChatGPT (OpenAI, 2023b), PaLM (Chowdhery et al., 2022), and LLaMA (Touvron et al., 2023), have demonstrated human-like capabilities to generate high-quality text, follow instructions, and respond to user queries. Although LLMs can improve the work efficiency of humans, they also pose several ethical and safety concerns, such as it becomes hard to differentiate LLM-generated text from human-written text. For example, LLMs may be inappropriately used for academic plagiarism or creating misinformation at large scale (Zellers et al., 2019). Therefore, it is important to de-

velop reliable approaches to protecting LLMs and detecting the presence of AI-generated texts, to mitigate the abuse of LLMs.

Toward this end, previous work has developed methods for automatically detecting text generated by LLMs. Existing methods mainly fall into three categories: 1) Classifier-based detectors by training a classifier, often a neural network, from data with AI-generated/human-written labels (Solaiman et al., 2019; OpenAI, 2023a); 2) Watermarking (Kirchenbauer et al., 2023) by injecting patterns into the generation of LLMs such that the pattern can be statistically detected but imperceptible to humans; 3) Likelihood-based detectors, e.g., DetectGPT (Mitchell et al., 2023), by leveraging the log-likelihood of generated texts. However, as recent research demonstrates that text classifiers are vulnerable to adversarial attacks (Iyyer et al., 2018; Ribeiro et al., 2018; Alzantot et al., 2018), these LLM text detectors may not be reliable when faced with adversarial manipulations of AI-generated texts.

In this paper, we stress-test the reliability of LLM text detectors. We assume that there is an LLM G that generates an output $\mathbf{Y} = G(\mathbf{X})$ given input \mathbf{X} . G is *protected* when there exists a detector f that can detect text normally generated by G with high accuracy. An *attack* aims to manipulate the generation process such that a new output \mathbf{Y}' is still plausible given input \mathbf{X} while the detector fails to identify \mathbf{Y}' as LLM-generated. The attack may leverage another attacker LLM G' .

In this context, we propose two novel attack methods. In the first method, we prompt G' to generate candidate substitutions of words in \mathbf{Y} , and we then choose certain substitutions either in a query-free way or through a query-based evolutionary search (Alzantot et al., 2018) to attack the detector. Our second method focuses on classifier-based detectors for instruction-tuned LLMs such as ChatGPT (OpenAI, 2023b). We automatically search for an additional instructional

*Alphabetical order. Authors contributed equally.

prompt with a small subset of training data for a given classifier-based detector. At inference time, the additional instructional prompt instructs the LLM to generate new texts that are hard to detect.

Several concurrent studies (Sadasivan et al., 2023; Krishna et al., 2023) proposed to attack detectors by paraphrasing AI-generated texts, with a different language model G' for paraphrasing. However, they assume that G' here is *not protected* by a detector. Paraphrasing with G' has been shown as effective in attacking detectors designed for the original LLM G , but it can become much less effective when G' is also protected by a detector since the paraphrased model can still be detected by the detectors of G' . In contrast, we demonstrate that even when the attacker LLM G' is also *protected* by a detector, we can still leverage G' for attacking LLM detectors. Therefore, even if all the strong LLMs are protected in the future, the currently existing detectors can still be vulnerable to our attacks.

We experiment with our attacks on all three aforementioned categories of LLM detectors. Our results reveal that all the tested detectors are vulnerable to our proposed attacks. The detection performance of these detectors degrades significantly under our attacks, while the texts produced by our attacks still mostly maintain reasonable quality as verified by human evaluation. Our findings suggest the current detectors are not sufficiently reliable yet and it requires further efforts to develop more robust LLM detectors.

2 Related Work

Detectors for AI-generated Text. Recent detectors for AI-generated text mostly fall into three categories. First, classifier-based detectors are trained with labeled data to distinguish human-written text and AI-generated text. For example, the AI Text Classifier developed by OpenAI (OpenAI, 2023a) is a fine-tuned language model. Second, watermarking methods introduce distinct patterns into AI-generated text, allowing for its identification. Among them, Kirchenbauer et al. (2023) randomly partition the vocabulary into a greenlist and a redlist during the generation, where the division is based on the hash of the previously generated tokens. The language model only uses words in the greenlists, and thereby the generated text has a different pattern compared to human-written text which does not consider such green-

lists and redlists. Third, DetectGPT (Mitchell et al., 2023) uses the likelihood of the generated text for the detection, as they find that text generated by language models tends to reside in the negative curvature region of the log probability function. Consequently, they define a curvature-based criterion for the detection.

Methods for Red-teaming Detectors. As the detectors emerge, several concurrent works showed that the detectors may be evaded to some extent, typically by paraphrasing the text (Sadasivan et al., 2023; Krishna et al., 2023). However, they need additional paraphrasing models which are typically unprotected models that are much weaker than the original LLM. Besides paraphrasing, Kirchenbauer et al. (2023) also discussed attacks against watermark detectors with word substitutions generated by a masked language model such as T5 (Raffel et al., 2020) which is a relatively weaker language model and tends to generate results with lower quality, and thus it may generate attacks with lower quality. On the other hand, Chakraborty et al. (2023) analyzed the possibilities of the detection given sufficiently many samples.

Adversarial Examples in NLP. Word substitution is a commonly used strategy in generating textual adversarial examples (Alzantot et al., 2018; Ren et al., 2019; Jin et al., 2020). Language models such as the BERT (Devlin et al., 2019) have also been used for generating word substitutions (Shi and Huang, 2020; Li et al., 2020; Garg and Ramakrishnan, 2020). In this work, we demonstrate the effectiveness of using the latest LLMs for generating high-quality word substitutions, and our query-based word substitutions are also inspired by the genetic algorithm in Alzantot et al. (2018) and Yin et al. (2020). For our instructional prompt, it is relevant to recent works that prompt LLMs to red team LLMs themselves (Perez et al., 2022) rather than detectors in this work. In addition, we fix a single instructional prompt at test time, which is partly similar to universal triggers in adversarial attacks (Wallace et al., 2019; Behjati et al., 2019), but unlike them constructing an unnatural sequence of tokens as the trigger, our prompt is natural and it is added to the input for the generative model rather than the detector directly.

Safety of Large Language Models. Detecting AI-generated texts is also related to the broader

Attack	Perturbation type	Test-time Queries		Applicability		
		G'	f	Classifier	Watermarking	Likelihood
Query-free Word Substitutions	Output	✓	—	✓	✓	✓
	Output	✓	✓	✓	—	✓
	Input	—	—	✓	—	—

Table 1: Properties of various attack methods and their applicability to various detectors. “Test-time queries” indicates whether each method requires querying G' or f for multiple times at test time.

topic of LLM safety. Research for the safety of LLMs aims to reduce privacy leakage and intellectual property concerns (Wallace et al., 2020; Carlini et al., 2021; Jagielski et al., 2023; Zhao et al., 2023), detect potential misuse (Hendrycks et al., 2018; Perez et al., 2022), defend against malicious users or trojans (Wallace et al., 2019, 2021), or detecting hallucinations (Zhou et al., 2021; Liu et al., 2022). See Hendrycks et al. (2021) for a roadmap of machine learning safety challenges. We test the reliability of some LLM text detection systems, which helps better understand the current progress in LLM text detection.

3 Settings and Overview

We consider a large language model G that conditions on an input context or prompt \mathbf{X} and generates an output text $\mathbf{Y} = G(\mathbf{X})$. We use upper-case characters to denote a sequence of tokens. For example, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$, where m is the sequence length. The model G is protected by a detector $f(\mathbf{Y}) \in [0, 1]$ that predicts whether \mathbf{Y} is generated by an LLM, where a higher detection score $f(\mathbf{Y})$ means that \mathbf{Y} is more likely to be LLM-generated. We use τ to denote a detection threshold such that \mathbf{Y} is considered LLM-generated if $f(\mathbf{Y}) \geq \tau$.

In this work, we consider three categories of detectors: (1) classifier-based detectors, (2) watermark detectors, and (3) likelihood-based detectors. For classifier-based detectors, a text classifier $f(\mathbf{Y})$ is trained on a labeled dataset with G -generated and human-written texts. For watermark detectors, G is modified from a base generator G_0 with a watermarking mechanism W , denoted as $G = W(G_0)$, and a watermark detector $f(\mathbf{Y})$ is constructed to predict whether \mathbf{Y} is generated by the watermarked LLM G . Specifically, we consider the watermarking mechanism in Kirchenbauer et al. (2023). For likelihood-based detectors, they estimate $f(\mathbf{Y})$ by comparing the

log probabilities of \mathbf{Y} and several random perturbations of \mathbf{Y} . Specifically, we consider Detect-GPT (Mitchell et al., 2023). We consider a model G as *protected* if there is a detector $f(\mathbf{Y})$ in place to protect the model from inappropriate usage.

To stress test the reliability and robustness of those detectors in this setting, we develop red-teaming techniques to generate texts that can downgrade a detector using an LLM that is also protected by this detector. We consider attacks by output perturbation and input perturbation respectively:

- **Output perturbation** perturbs the original output \mathbf{Y} and generates a perturbed output \mathbf{Y}' .
- **Input perturbation** perturbs the input \mathbf{X} into \mathbf{X}' as the new input, leading to a new output $\mathbf{Y}' = G(\mathbf{X}')$.

In both cases, we aim to minimize $f(\mathbf{Y}')$ so that the new output \mathbf{Y}' is wrongly considered as human-written by the detector f . Meanwhile, we require that \mathbf{Y}' has a quality similar to \mathbf{Y} and remains a plausible output to the original input \mathbf{X} . For our attack algorithms, we also assume that the detector f is black-box—only the output scores are visible but not its internal parameters.

We propose to attack the detectors in two different ways. In Section 4, we construct an output perturbation by replacing some words in \mathbf{Y} , where we prompt a protected LLM G' to obtain candidate substitution words, and we then build query-based and query-free attacks respectively to decide substitution words. In Section 5, if G is able to follow instructions, we search for an instructional prompt from the generation by G and append the prompt to \mathbf{X} as an input perturbation, where the instructional prompt instructs G to generate texts in a style making it hard for the detector to detect. Table 1 summarizes our methods and their applicability to different detectors. At

test time, instructional prompts are fixed and thus totally query-free. For word substitutions, they require querying G' multiple times to generate word substitutions on each test example; the query-free version does not repeatedly query f while the query-based version also requires querying f multiple times. In practice, we may choose between these methods depending on the query budget and their applicability to the detectors.

4 Attack with Word Substitutions

To attack the detectors with output perturbations, we aim to find a perturbed output \mathbf{Y}' that is out of the original detectable distribution. This is achieved by substituting certain words in \mathbf{Y} . To obtain suitable substitution words for the tokens in \mathbf{Y} that preserve the fluency and semantic meaning, we utilize a protected LLM denoted as G' . For each token in \mathbf{Y} denoted as \mathbf{y}_k , we use $s(\mathbf{y}_k, \mathbf{Y}, G', n)$ to denote the process of generating at most n word substitution candidates for \mathbf{y}_k given the context in \mathbf{Y} by prompting G' , and $s(\mathbf{y}_k, \mathbf{Y}, G', n)$ outputs a set of at most n words. Note that not every word can be substituted, and $s(\mathbf{y}_k, \mathbf{Y}, G', n)$ can be an empty set if it is not suitable to replace \mathbf{y}_k . We will discuss how we generate the word substitution candidates using G' in Section 4.1.

General Attack Objective. The objective of attacking f with word substitutions can be formulated as a minimization problem given a substitution budget ϵ :

$$\begin{aligned} \mathbf{Y}' &= \arg \min_{\mathbf{Y}'} f(\mathbf{Y}'), \quad (1) \\ \text{s.t. } \mathbf{y}'_k &\in \{\mathbf{y}_k\} \cup s(\mathbf{y}_k, \mathbf{Y}, G', n), \\ &\sum_{k=1}^m \mathbb{1}(\mathbf{y}_k \neq \mathbf{y}'_k) \leq \epsilon m. \end{aligned}$$

Here we aim to find an optimally perturbed output \mathbf{Y}' that minimizes the predicted score $f(\mathbf{Y}')$ among all possible \mathbf{Y}' . Each word in the perturbed output \mathbf{y}'_k is either the unperturbed word \mathbf{y}_k or selected from the word substitution candidates $s(\mathbf{y}_k, \mathbf{Y}, G', n)$, and the total number of perturbed words is at most ϵm . To solve the minimization problem in Eq. (1), we consider both query-free and query-based substitutions respectively. We may choose between the two methods

ChatGPT

Given this sentence: “*The scientists said the findings could help improve treatments and lead to new ways of preventing breast cancer spreading. Experts have described the findings as “exciting”.*”, **for each word** in “*findings, exciting*”, **give 10 substitution words** that **do not change the meaning of the sentence**. **Return each word and its substitutions in one line, in the format of “word:substitutions”**

findings: results, discoveries, outcomes, conclusions, observations, data, evidence, findings, research, studies
exciting: thrilling, exhilarating, stimulating, electrifying, gripping, captivating, invigorating, inspiring, breathtaking, exciting

LLaMA

“*The scientists said the findings could help improve treatments and lead to new ways of preventing breast cancer spreading. Experts have described the findings as “exciting”.*”

Synonyms of the word “exciting” in the above sentence are:

- a) “interesting”
- b) “surprising”
- c) “unusual”

Table 2: Prompts for generating word substitution candidates using ChatGPT and LLaMA as well as the corresponding outputs. Text in bold denotes the prompt template. Text in italic denotes a text to be perturbed or words to be replaced for a given example. The generated word substitutions are in blue and listed after the bold text.

depending on whether the attacker can query f multiple times.

4.1 Generating Word Substitution Candidates

Table 2 shows the prompts we use and the outputs produced by G' , when G' is ChatGPT and LLaMA, respectively. ChatGPT is able to follow instructions, and thus our prompt is an instruction asking the model to generate substitution words, and multiple words can be substituted simultaneously. For LLaMA, which has less instruction-following ability, we expect it to generate a text completion following our prompt, where the prompt is designed such that a plausible text completion consists of suggested substitution words, and we replace one word at a time.

The benefit of applying an LLM here is that it enables us to obtain substitution words that not only have similar meanings with the original word but are also compatible with the context, as previous works also used language models such as BERT for generating adversarial examples (Shi and Huang, 2020). Thus it is more convenient than earlier methods using synonym lists for generating substitution words which need to be further checked with a separate language model (Alzantot et al., 2018) for compatibility with the context.

4.2 Query-based Word Substitutions

For query-based substitutions, we use the evolutionary search algorithm (Alzantot et al., 2018; Yin et al., 2020) originally designed for generating adversarial examples in NLP. The algorithm starts from a *population* of perturbed texts which includes input texts with a certain amount of tokens randomly replaced. Then, it iterates over several *generations* of populations to select elites in each population, i.e., the most effective substitution that leads to the lowest detection score. New generations are constructed by crossing over the elite substitutions in the previous generation.

4.3 Query-free Word Substitutions

For the query-free attack, we simply apply word substitution on random tokens in \mathbf{Y} to attack DetectGPT and classifier-based detectors. For watermark detectors, we further design an effective query-free attack utilizing the properties of the detection method.

Specifically, for the watermark in Kirchenbauer et al. (2023), the watermarked LLM generates a token by modifying the predicted logits at position $i+1$: $g(\mathbf{y}_{i+1} \mid [\mathbf{y}_1, \dots, \mathbf{y}_i]) = g_0(\mathbf{y}_{i+1} \mid [\mathbf{y}_1, \dots, \mathbf{y}_i]) + \delta$ if the candidate token \mathbf{y}_{i+1} is in the greenlist, where we use g_0 to denote the output logits of the original model G_0 and g for the watermarked model G . δ is an offset value for shifting the logits of greenlist tokens, and γ is the proportion of greenlist tokens in the vocabulary. Therefore, a text generated by the watermarked model tends to have more greenlist tokens compared to a text generated by the original model. $f(\mathbf{Y})$ calculates the detection score based on the number of greenlist tokens in \mathbf{Y} as:

$$f(\mathbf{Y}) = (|s_G| - \gamma T) / \sqrt{T\gamma(1 - \gamma)}, \quad (2)$$

where $|s_G|$ is the number of greenlist tokens in \mathbf{Y} and T is the total number of tokens in \mathbf{Y} .

Therefore, given a fixed substitution budget ϵ , we aim to identify and substitute more greenlist tokens to reduce the total count of greenlist tokens. We achieve this with a two-stage algorithm. At the first stage, we sort all tokens in \mathbf{Y} by the prediction entropy estimated with a language model M , which can be either the same generative model G or a weaker model as we only use the entropy as a heuristic score. The prediction entropy is estimated by feeding M with the prefix or masked text without the word to be estimated. As the watermarking offset δ is applied on the decoding process, a token with higher entropy is easier to be affected by watermarking. At the second stage, we pick ϵm tokens with highest entropy and use a watermarked LLM G' to generate word substitutions as introduced in Section 4.1.

5 Attack by Instructional Prompts

In this section, we build attacks by perturbing the input prompt to encourage LLMs to generate texts that are difficult to detect. In particular, we focus on LLM-based generative models that can follow instructions and classifier-based detectors. We consider ChatGPT (OpenAI, 2023b) as the generative model G and OpenAI AI Text Classifier (OpenAI, 2023a) as the detector f . The OpenAI AI Text Classifier is a fine-tuned neural network, while neural networks have been shown to be vulnerable to distribution shifts in NLP literature (Miller et al., 2020; Awadalla et al., 2022). Therefore, we aim to shift the generated text to a different distribution where the detector is more likely to fail. We do not require the shifted generation to be semantically equivalent to the original text, but the generation should still be a plausible output to the given input.

We achieve this by searching for an additional prompt \mathbf{X}_p appended to the original input \mathbf{X} , which forms a new input $\mathbf{X}' = [\mathbf{X}, \mathbf{X}_p]$ to G . In particular, \mathbf{X}_p consists of \mathbf{X}_{ins} and \mathbf{X}_{ref} , where \mathbf{X}_{ins} is an instruction asking the model to follow the writing style of reference \mathbf{X}_{ref} .

Searching for \mathbf{X}_p . We search for \mathbf{X}_p on a small subset of training examples with n examples $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$. We assume that we can query the detector f for multiple times during search time. After an effective \mathbf{X}_p is found, it can be

Generative Model	DetectGPT	Watermarking	Classifier-based Detector
GPT-2-XL	ChatGPT	LLaMA-65B	ChatGPT
LLaMA-65B	LLaMA-65B	LLaMA-65B	–
ChatGPT	ChatGPT	–	ChatGPT

Table 3: The protected LLM G' used in generating perturbations for each generative model G and the detectors. “–” indicates a combination of the generative model and the detector is not applicable.

applied universally on all inputs from this dataset at test time. The objective of the search is:

$$\arg \min_{\mathbf{X}_p} \frac{1}{n} \sum_{i=1}^n \mathbb{1}(f(G([\mathbf{X}_i, \mathbf{X}_p])) \geq \tau), \quad (3)$$

which aims to minimize the average detection rate for the new outputs generated with \mathbf{X}_p appended to the input.

We use G to generate various \mathbf{X}_{ins} and \mathbf{X}_{ref} in each iteration and try to search for an optimal $\mathbf{X}_p = [\mathbf{X}_{\text{ins}}, \mathbf{X}_{\text{ref}}]$ following the objective in Eq. (3). Initially, we set \mathbf{X}_{ins} to a manually written instruction, “Meanwhile please imitate the writing style and wording of the following passage:”. An initial value for \mathbf{X}_{ref} is not necessary. We also create and initialize a priority queue \mathcal{O} with n initial outputs generated from the n training examples without \mathbf{X}_p . \mathcal{O} sorts its elements according to the detection scores from f and prioritize those with lower scores. In each iteration of the search, we have two steps:

- Updating \mathbf{X}_{ref} : We pop the top- K candidates from \mathcal{O} . For each candidate, we combine it with the current \mathbf{X}_{ins} respectively as the potential candidates for \mathbf{X}_p in the current iteration.
- Updating \mathbf{X}_{ins} : We instruct model G to generate K variations of the current \mathbf{X}_{ins} , inspired by Zhou et al. (2023) for automatic prompt engineering. And we combine them with the current \mathbf{X}_{ins} respectively as the potential candidates for \mathbf{X}_p .

For both of these two steps, we take the best candidate \mathbf{X}_p according to Eq. (3). When generating $G([\mathbf{X}_i, \mathbf{X}_p])$ in Eq. (3), we push all the generated outputs to \mathcal{O} as the candidates for \mathbf{X}_{ref} in the later rounds. We take T iterations and return the

final $\mathbf{X}_p = [\mathbf{X}_{\text{ins}}, \mathbf{X}_{\text{ref}}]$ to be used at test time, and $\mathbf{Y}' = G([\mathbf{X}, \mathbf{X}_p])$ is the new output given input \mathbf{X} .

For some \mathbf{X}_p , we find that the G may directly copy text from \mathbf{X}_{ref} to generate \mathbf{Y}' when \mathbf{X}_p is appended into the input prompt. To prevent this behavior, we compute a matching score between \mathbf{X}_{ref} and \mathbf{Y}' and discard a candidate \mathbf{X}_p during the search if more than 20% of words from \mathbf{X}_{ref} (except stop words) appear in \mathbf{Y}' . In this way, we find that the copying behavior is effectively prevented.

6 Experiments

6.1 Experimental Settings

Generative Models and Detectors. We experiment with a wide range of generative LLMs and corresponding detectors. For the generative model G , we consider GPT-2-XL (Radford et al., 2019), LLaMA-65B (Touvron et al., 2023), and ChatGPT (gpt-3.5-turbo) (OpenAI, 2023b). For detectors, we consider DetectGPT (Mitchell et al., 2023), watermarking (Kirchenbauer et al., 2023), and classifier-based detectors (OpenAI, 2023a; Solaiman et al., 2019). For DetectGPT, we use GPT-Neo (Black et al., 2021) as the scoring model to estimate the log-likelihood. DetectGPT also requires masking spans of the texts and filling in the spans with an external T5-3B model (Raffel et al., 2020). We fix the mask rate to be 15%. Watermarking is applied to open-source LLaMA-65B and GPT-2-XL but not ChatGPT, as it requires logits scores in generation. We use $\gamma = 0.5$ in all the watermarking experiments, following the default setting in Kirchenbauer et al. (2023). Moreover, classifier-based detectors include a fine-tuned RoBERTa-Large detector (Solaiman et al., 2019) for GPT-2 texts and the OpenAI AI Text Classifier (OpenAI, 2023a) for ChatGPT texts. We summarize all generative models and detectors considered in the experiments in Table 3.

Generative Model	Dataset	Unattacked	Dipper Paraphrasing	Query-free Substitution	Query-based Substitution
GPT-2-XL	XSum	84.4	35.2	25.9	3.9
	ELI5	70.6	36.7	21.2	3.8
ChatGPT	XSum	56.0	34.6	25.6	4.5
	ELI5	55.0	39.5	12.2	6.5
LLaMA-65B	XSum	59.3	49.0	25.5	9.9
	ELI5	60.5	53.1	31.4	18.6

Table 4: AUROC scores (%) of DetectGPT under various attack settings.

For experiments involving watermarking, we use a watermarked LLaMA-65B as G' , as we cannot implement watermarking on ChatGPT; we also use LLaMA-65B as G' in the setting with LLaMA-65B itself as G and DetectGPT as the detector; and in other settings, we use ChatGPT as G' which is protected by either DetectGPT or the classifier-based detector. The choice of protected LLM G' is also summarized in Table 3.

Baselines. To demonstrate the advantage of our methods in revealing detectors’ weakness, we compare with several baselines. Dipper paraphrase (Krishna et al., 2023) is a recent method that trains a paraphrasing model to rewrite AI texts and bypass detectors. It prepends diversity codes to control the level of paraphrases introduced to the texts. We use Dipper-paraphraser-XXL with 20 lexical diversity and 60 order diversity to paraphrase the AI texts, which keeps the same level of 20% uni-gram difference as word substitution. We use nucleus sampling (Holtzman et al., 2019) with $p = 0.9$ for the paraphraser. And in ChatGPT experiments, we also use ChatGPT itself for paraphrasing.

Datasets. We mainly use two types of datasets including text completion and long-form question answering. We use XSum (Narayan et al., 2018) for text completion, where we take the first sentence as the input prompt for the completion, and we use ELI5 (Fan et al., 2019) for long-form question answering. In addition, for the RoBERTa-Large detector, we also use a specific GPT-2 output dataset (Solaiman et al., 2019) as they are fine-tuned solely on GPT-2 texts. Since the OpenAI AI Text Classifier requires the text to contain at least 1,000 characters, we filter XSum and ELI5 datasets and only retain examples with human reference text containing at least 1,000 characters.

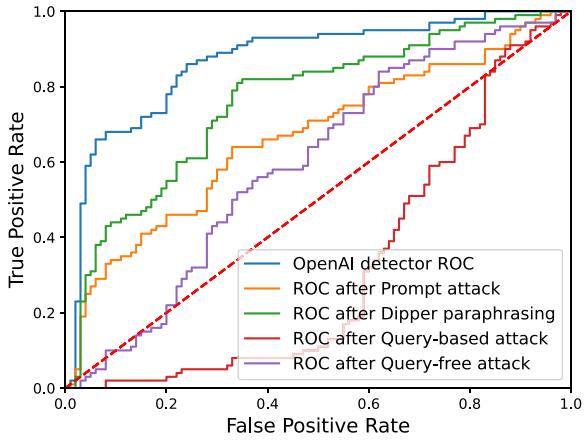


Figure 1: ROC plot of OpenAI AI Text Classifier under different attack methods. We show the ROC plot on the ELI5 dataset in Table 10.

For each dataset, we shuffle the test set and use the first 100 examples.

Metrics. We use several metrics to evaluate the detectors under attacks. Area Under the Receiver Operating Characteristic Curve (AUROC) scores summarize the performance of detectors under various thresholds. A detection rate (DR) is the true positive rate under a fixed threshold (positive examples mean LLM-generated texts), where we either tune the threshold to meet a particular false positive rate or follow the original thresholds of the detectors. For the GPT-2 output dataset, we also use Attack Success Rate (ASR) which computes the rate that the attack successfully flips the prediction by the detector, out of all the positive examples on which the detector originally predicts correctly.

6.2 Attack with Word Substitutions

We apply word substitution-based attack on all the three categories of detection methods. In each

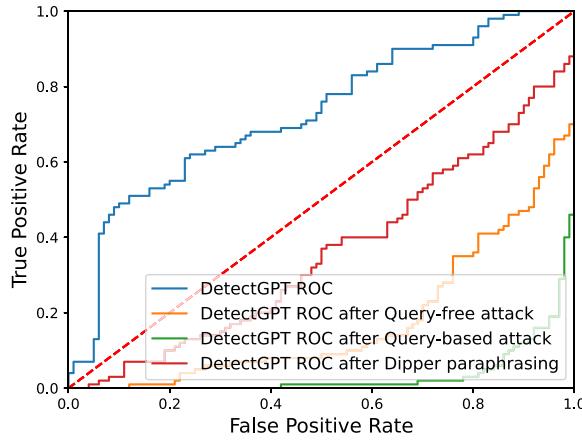


Figure 2: ROC plot of DetectGPT detectors under different attack methods. We show the ROC plot on the ELI5 dataset in Table 4 for the GPT2-XL model.

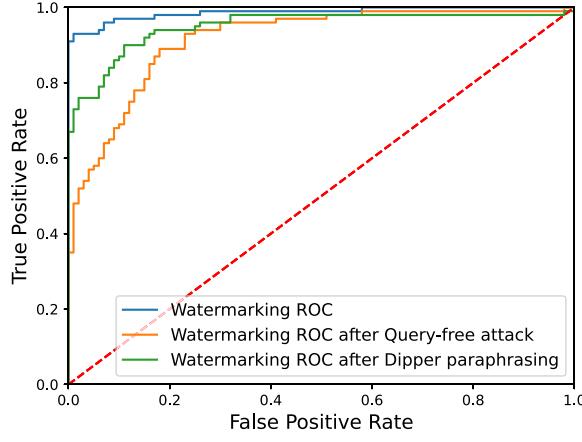


Figure 3: ROC plot of watermark detectors under different attack methods. We show the ROC plot on the ELI5 dataset in Table 5 for the GPT2-XL model with $\delta = 1.5$.

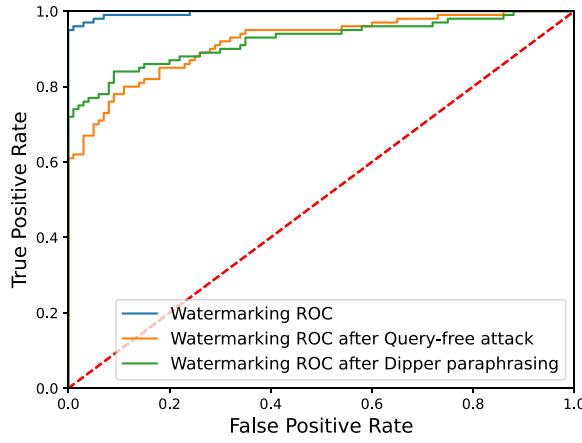


Figure 4: ROC plot of watermark detectors under different attack methods. We show the ROC plot on the ELI5 dataset in Table 5 for the LLaMA-65B model with $\delta = 1.5$.

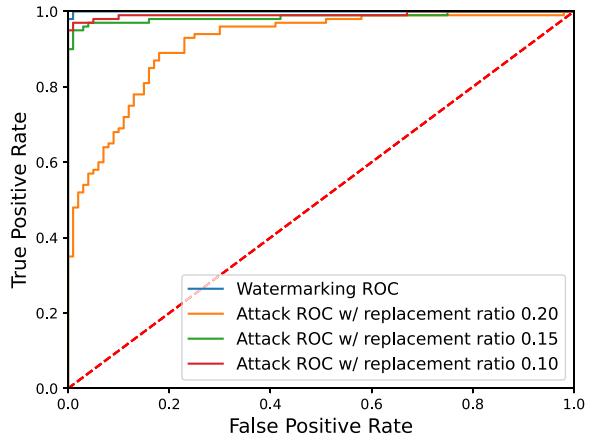


Figure 5: ROC plot of watermark detectors under query-free attacks with different replacement ratios. We show the ROC plot on the ELI5 dataset for the GPT2-XL model with $\delta = 1.5$.

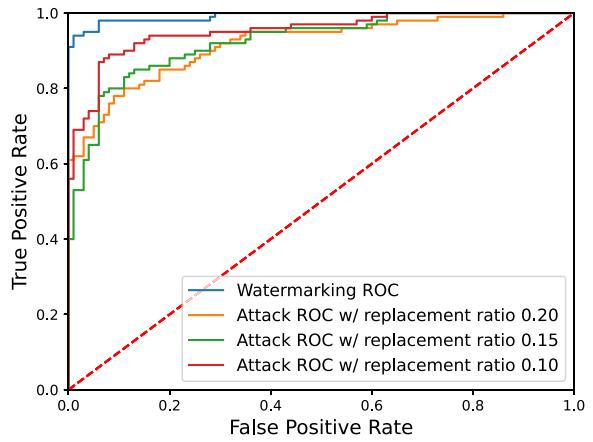


Figure 6: ROC plot of watermark detectors under query-free attacks with different replacement ratios. We show the ROC plot on the ELI5 dataset for the LLaMA-65B model with $\delta = 1.5$.

setting, we assume that both G and G' are protected by the same detector f . We limit the number of substituted words to be at most 20% of the total number words except stop words and proper nouns that should not be substituted. This leads to around 7 substituted tokens per example. For the evolutionary search, it requires 70 queries per example with a population size of 10.

Attack against DetectGPT The results are shown in Table 4. We show an example of ROC plot in Figure 2. We find that DetectGPT’s AUROC drops below random guess when we simply apply query-free substitutions which randomly select substitutions from the candidate pool. For example, on XSum, AUROC on GPT-2-XL

Generative Model	δ	Dataset	Unattacked		Dipper Paraphrasing		Query-free Substitution	
			AUROC	DR	AUROC	DR	AUROC	DR
GPT-2-XL	1.0	XSum	98.0	81.0	84.7	38.0	85.9	34.0
		ELI5	97.14	72.0	86.3	34.0	86.9	33.0
	1.5	XSum	99.4	96.0	94.1	72.0	91.6	35.0
		ELI5	98.7	91.0	95.0	67.0	91.5	47.0
LLaMA-65B	1.0	XSum	88.9	22.0	79.3	12.0	67.8	10.0
		ELI5	94.7	70.0	81.6	45.0	79.6	34.0
	1.5	XSum	96.4	63.0	85.5	27.0	84.5	16.0
		ELI5	99.6	95.0	92.3	72.0	91.8	61.0

Table 5: Attack against the watermark detector. We report both AUROC scores (%) and the detection rates (DR) (%). For DR, we set the decision threshold such that the false positive rate for the human reference text on the same test examples is 1%.

drops from 84.4% to 25.9%. For ChatGPT and LLaMA-65B, AUROC drops from 56.0% to 25.6% and from 59.3% to 25.5%, respectively. The AUROC scores further drop to only 3.9%, 4.5%, and 9.9% respectively with the query-based evolutionary search. Our word substitution methods consistently surpass the Dipper paraphrasing, which demonstrates that our methods are revealing more vulnerability of the detectors. We do not use DR here as we find that the DR values are already very low (usually below 10%) even when no attack is applied, and we follow Mitchell et al. (2023) which originally also only used AUROC.

Attack against Watermarking We use a T5-Large model to estimate the prediction entropy for each token. In our main experiments, we select 20% of tokens in the initial output \mathbf{Y} with highest prediction entropy to be replaced in the attack. We also compare the attack performance with different replacement ratios in Figure 5 and Figure 6. In all experiments in Table 5, we keep the first 100 tokens in each text. We filter the suggested word substitutions to keep fewer than 4 tokens in the substitution candidates to avoid invalid substitution. We report the AUROC score and detection rate for each setting in Table 5. We show two examples of ROC plot in Figure 3 and 4. For the detection rates, we set the threshold value to keep the false positive rate for human texts equal to 1% following Krishna et al. (2023). The results show that the detection rates are significantly degraded after the query-free word substitution attack under two different watermarking settings with $\delta = 1.0, 1.5$. Although the

Original output

The scientists said the findings could help improve treatments and lead to new ways of preventing breast cancer spreading. Experts have described the findings as “exciting”. Bone is the most common site for breast cancer to spread to. Once breast cancer reaches the bone, it can be treated but often is not curable. In experiments in mice, the Sheffield researchers found breast cancer cells were sending signals to the cells inside

Query-free Substitution

The **researchers** said the findings could help improve **drugs** and lead to new ways of **stopping** breast cancer **metastasizing**. Experts have **said** the discoveries as “**interesting**”. Bone is the most common **place** for **bone** cancer to **metastasize** to. Once **bone** cancer **spreads to** the bone, it can be treated but usually is not **can be treated**. In **research in the Sheffield researchers**, the Sheffield **Scientists** found **cancer** cancer cells were sending signals to the cells **in**.

Table 6: An example from the XSum dataset. We show the original output from watermarked LLaMA-65B, as well as the output after query-free word substitution attack.

detection rate on unattacked texts can be further increased by increasing δ , in practice, the watermark strength should be kept under an appropriate level to avoid hurting the quality of text generation (Kirchenbauer et al., 2023). Compared to Dipper paraphrasing, we achieve lower detection rates without using a separate unprotected paraphraser model. We also show qualitative examples for attacks against watermarking in Table 6.

Attack	ASR
Dipper Paraphrasing	4%
Query-free Substitutions	2%
Query-based Substitutions	68%

Table 7: Attack Success Rate (ASR) for OpenAI RoBERTa-Large detector for GPT-2 texts.

Initial prompt for querying ChatGPT on XSum
Please complete this passage with at least 150 words: {X}
Initial prompt for querying ChatGPT on ELI5
Please answer this question with at least 150 words: {X}
Prompt for paraphrasing
Please paraphrase the following passage, with at least 200 words: {Y}

Table 8: Prompts used for querying ChatGPT. Initial prompts are used for instructing ChatGPT to perform text completion or question answering on XSum and ELI5 respectively. And the prompt for paraphrasing is used in Table 10 for paraphrasing \mathbf{Y} into \mathbf{Y}' directly. We also instruct ChatGPT to generate at least 150 words as the OpenAI AI Text Classifier does not accept shorter texts.

X_p on XSum
During the waiting period, please take into consideration utilizing the writing style and vocabulary used in the subsequent paragraph.
“Wales football star, Gareth Bale, is set to undergo surgery on his ankle after suffering an injury during Real Madrid’s 2-1 victory over Sporting Lisbon in the Champions League. (. . .)”
X_p on ELI5
At the same time, kindly mimic the writing technique and diction utilized in the subsequent excerpt.
“The reason why metal feels cooler compared to other things at the same temperature is due to its thermal conductivity. (. . .)”

Table 9: Our searched instructional prompts on XSum and ELI5 respectively. Part of the \mathbf{X}_{ref} is omitted due to the space limit.

Attack against Classifier-based Detectors Results for attacking GPT-2 text detector are shown in Table 7. We find that the attack success rate (ASR) on detecting GPT-2 texts is close to 0 for

Method	XSum		ELI5	
	AUROC	DR	AUROC	DR
Unattacked	88.8	30.0	87.1	54.0
ChatGPT Paraphrasing	80.0	14.0	76.2	27.0
Query-free Substitution	69.9	2.0	59.0	2.0
Query-based Substitution	43.4	0.0	31.5	0.0
Instructional Prompts	54.9	5.0	66.7	21.0

Table 10: AUROC scores (%) and detection rates (DR) (%) of the OpenAI AI Text Classifier on the original outputs by ChatGPT and outputs with various attacks respectively.

Dataset	Method	Fluency	Plausibility
XSum	Unattacked	2.79±0.47	2.76±0.52
	ChatGPT Paraphrasing	2.60±0.52	2.75±0.43
	Query-free Substitution	2.58±0.57	2.68±0.50
	Query-based Substitution	2.49±0.67	2.68±0.52
	Instructional Prompts	2.67±0.57	2.66±0.55
ELI5	Unattacked	2.78±0.45	2.87±0.34
	ChatGPT Paraphrasing	2.35±0.48	2.95±0.28
	Query-free Substitution	2.83±0.37	2.60±0.49
	Query-based Substitution	2.65±0.54	2.63±0.47
	Instructional Prompts	2.73±0.50	2.56±0.62

Table 11: Average score and standard deviation of ratings from human evaluation on attacks against the OpenAI AI Text Classifier for ChatGPT.

Dataset	Method	Fluency	Plausibility
XSum	Unattacked	2.79±0.47	2.76±0.52
	Dipper Paraphrasing	2.67±0.51	2.60±0.52
	Query-free Substitution	2.58±0.57	2.68±0.50
	Query-based Substitution	2.35±0.51	2.40±0.49
ELI5	Unattacked	2.78±0.45	2.87±0.34
	Dipper Paraphrasing	2.63±0.48	2.68±0.47
	Query-free Substitution	2.83±0.37	2.60±0.49
	Query-based Substitution	2.47±0.50	2.42±0.53

Table 12: Average score and standard deviation of ratings from human evaluation on attacks against DetectGPT for detecting ChatGPT generation.

Dataset	Method	Fluency	Plausibility
XSum	Unattacked	2.65±0.51	2.68±0.53
	Dipper Paraphrasing	2.68±0.47	2.58±0.59
	Query-free Substitution	2.45±0.69	2.68±0.47
ELI5	Unattacked	2.45±0.50	2.42±0.49
	Dipper Paraphrasing	2.63±0.51	2.66±0.53
	Query-free Substitution	2.27±0.62	2.58±0.49

Table 13: Average score and standard deviation of ratings from human evaluation on attacks against the watermark detector for the watermarked LLaMA-65B with $\delta = 1.0$, $\gamma = 0.5$.

Original input

Please answer this question with at least 150 words:

Why we don't yet have a cable provider that will let me pick and pay for ONLY the channels I want?

Original output

The reason why we do not yet have a cable provider that will allow choosing and paying for only the channels we want is that it may not be economically feasible for the providers. The cable providers typically negotiate content deals with media companies in packages or bundles. This packaging provides providers with a lower cost per channel, making it more profitable for them. (. . .)

ChatGPT Paraphrasing

The lack of cable providers that offer the option to choose and pay for individual channels is due to the feasibility of the idea. Cable providers, such as media companies, negotiate bundles or packages of channels rather than individual offerings. This arrangement allows for a lower cost per channel for the providers, which translates to higher profits. Additionally, media companies benefit from bundling channels as it increases the exposure of lower-demand channels and ensures programming visibility. (. . .)

Query-free Substitution

The **motive** why we do not yet have a cable provider that will allow choosing and paying for only the channels we want is that it may not be **monetarily** feasible for the providers. The cable providers typically negotiate content deals with media companies in **wrap** or bundles. This packaging provides providers with a lower **expense** per channel, making it more profitable for them. . The media companies also benefit from **integrating** their channels, as they can sell lower-demand channels as a package deal, ensuring broader visibility for their programs. (. . .)

Query-based Substitution

The reason why we do not yet have a cable provider that will allow **nominating** and paying for only the channels we want is that it may not be economically feasible for the providers. The **Cablegram** providers typically negotiate content deals with media companies in packages or bundles. This **parcel** provides providers with a lower cost per channel, making it more profitable for them. The media companies also benefit from packaging their channels, as they can sell lower-demand **avenue** as a package deal, ensuring broader visibility for their programs. (. . .)

Instructional Prompts

The reason why we don't yet have a cable provider that lets us pick and pay for only the channels we want is due to the complex and structured industry model that cable companies have established. The traditional cable model focuses on bundling channels together into packages, often forcing customers to pay for channels they don't want or need. This model benefits cable companies by allowing them to earn more revenue from customers who are willing to pay for premium packages, even if they don't watch all the channels that come with it. (. . .)

Table 14: An example from the ELI5 dataset when using ChatGPT as the generative model. We show the original input and output, as well as the output under various attacks. Due to space limit, we omit part of the generation as indicated by “(. . .)”.

both paraphrasing and query-free substitutions. We hypothesize that this is because the detector is specifically trained on detecting GPT-2 texts, and it is hard to remove the patterns leveraged by those detectors by randomly selecting word substitutions or paraphrasing. Our evolutionary search-based substitutions achieve much better ASR compared to the query-free methods.

For the OpenAI AI Text Classifier shown in Table 10, query-free attacks are able to decrease the detection AUROC by 18.9 and 28.1 percentage points on XSum and ELI5, respectively, while query-based ones further decrease them by 45.4 and 55.6 percentage points to lower than ran-

dom. Comparison with the attack using instructional prompts and more details are discussed in Section 6.3.

6.3 Attack with Instructional Prompts

We conduct experiments for our instructional prompts using ChatGPT as the generative model and the OpenAI AI Text Classifier as the classifier-based detector. The detector is `model-detect-v2` accessible via OpenAI APIs as of early July 2023. We choose this detector as it is developed by a relatively renowned company and has been shown to achieve stronger detection accuracy (Krishna et al., 2023) than other

classifier-based detectors such as GPTZero (Tian and Cui, 2023). This detector was also available at no cost when our experiments were conducted. Its output contains five classes, including “likely”, “possibly”, “unclear if it is”, “unlikely” and “very unlikely”, with thresholds 0.98, 0.90, 0.45, and 0.10 respectively. We follow these thresholds and use a threshold of 0.9 to compute detection rates.

We search for the instructional prompt using $n = 50$ training examples, $T = 5$ iterations, and $K = 5$ candidates in each iteration. We show the prompts for querying ChatGPT in Table 8 and the results in Table 10 and Figure 1. Our instructional prompts significantly reduce the the AUROC scores and detection rates compared to the unattacked setting and are more effective than paraphrasing with ChatGPT. While using instructional prompts may not lead to lower AUROC or DR compared to word substitutions, it does not require querying G' or f multiple times, making it a more efficient and equally effective option. We show an example on ELI5 with various attacks in Table 14 and the instructional prompts found by our algorithm in Table 9.

7 Human Evaluation

To validate that our approach mostly preserves the quality of the generated text, we conduct a human evaluation on Amazon Mechanical Turk (MTurk). On each dataset, we consider the first 20 test examples and ask 3 MTurk workers to rate the quality of the text generated by each method on each of the test examples. Specifically, we use two metrics, including fluency and plausibility, where fluency measures whether the text is grammatically correct and fluent, and plausibility measures whether the generated text is a plausible output given the input, on either the text completion (XSum) or long-form question answering (ELI5) task. We use a 1/2/3 rating scale for each of the metrics (3 is the best and vice versa), and we provide the workers with guidance on the ratings, according to whether there are many/several/ almost no issues for the 1/2/3 ratings on fluency and plausibility respectively. The workers are paid USD \$0.05 for each example and we provide an additional bonus. The annotation time varies, but the estimated wage rate is \$10/hr, which is higher than the US minimum wage (\$7.25/hr).

Tables 11 to 13 show results on attacks against the three detectors respectively. These results show that our attack methods can maintain reasonable and satisfactory plausibility and fluency with a small degradation compared to the unattacked texts. Among our attack methods, we find that the query-free substitution usually has better fluency and also sometimes better plausibility compared to the query-based substitution, as the query-based one which aims to search for a stronger attack tends to degrade the text quality slightly more. Our method with instructional prompts has better fluency than the query-based substitution and sometimes better fluency than the query-free substitution, and its generation is directly from model G without further substituting words; it also has comparable plausibility compared to the word substitution methods.

8 Discussions

Robustness of Detectors. Comparing the results in Tables 4, 5, and 10, we see a clear trend that watermarking is relatively more robust to the attacks compared to the other two techniques. The detection mechanism in watermarking is mainly a statistical method and tends to be more robust compared to the likelihood-based and classifier-based detectors which heavily rely on neural networks. However, watermarking is also the only method here that modifies the generation process of the protected LM. It requires access to the intermediate outputs of the LM and the generation quality may degrade. While the three detectors are not strictly comparable, as watermarking has a different setting by modifying the generation, our results still show insights on the different degrees of robustness of the various detectors. From Table 4 and Table 10, we can also see that query-based method generally produces stronger attacks, which benefits from the guidance of multiple queries to the detector when searching for more effective word substitutions.

Text Quality under the Attacks. We note that attack approaches often result in a minor decline in generation performance. Nonetheless, based on human assessment, the quality remains acceptable and is adequate to spam the target in a real-world scenario. Note that in practical scenarios such as online spamming, malicious actors do not have to use perfect text and they may still use text with

slightly degraded quality, as their main purpose is not to generate perfect text but text that is hard to detect. Therefore, the insufficient robustness of existing detection strategies continues to be a significant concern.

Defending against the Attacks. Inspired by our attack results, we discuss on potential directions for developing methods to defend against the attacks. One possibility is to combine watermarking with a likelihood estimation to defend against word substitution attacks. This is based on the observation that the word substitution attacks often need to substitute around 20% tokens from greenlisted tokens to redlisted tokens. After the word substitution, the new redlisted tokens tend to have lower probabilities in the prediction by the original watermarked model, and the new text also tends to have a higher perplexity under the watermarked model. Thus, one may leverage a watermarked language model to check the perplexity or the likelihood of all the redlist tokens, to predict whether a word substitution attack is possibly present.

9 Conclusion and Limitations

In this work, we study the reliability of three distinct types of LLM text detectors by proposing two attack strategies: 1) word substitutions and 2) instructional prompts using protected LLMs. Experiments reveal the vulnerability of existing detectors, which urges the design of more reliable LLM text detectors. We have released the source code and data with BSD-3-Clause license at GitHub (<https://github.com/shizhouxing/LLM-Detector-Robustness>).

Finally, the purpose of this work is to test and reveal the limitations of the currently existing LLM text detectors, and we red-team the detectors for future works to improve their robustness and reliability based on our proposed evaluation. Thus this work is potentially beneficial to for developing future systems protecting LLMs and preventing abusive usage. The proposed approaches should not be used to bypass real-world LLM text detectors for malicious purposes.

Acknowledgments

We thank UCLA-NLP, the action editor, and the reviewers for their invaluable feedback. The work

is supported in part by CISCO, NSF 2008173, 2048280, 2325121, 2331966, ONR N00014-23-1-2300:P00001, and DARPA ANSR FA8750-23-2-0004. KC was supported as a Sloan Fellow.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. <https://doi.org/10.18653/v1/D18-1316>

Anas Awadalla, Mitchell Wortsman, Gabriel Ilharco, Sewon Min, Ian Magnusson, Hannaneh Hajishirzi, and Ludwig Schmidt. 2022. Exploring the landscape of distributional robustness for question answering models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5971–5987. <https://doi.org/10.18653/v1/2022.findings-emnlp.441>

Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. 2019. Universal adversarial attacks on text classifiers. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7345–7349. IEEE. <https://doi.org/10.1109/ICASSP.2019.8682430>

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. *Zenodo*. <https://doi.org/10.18653/v1/2022.bigscience-1.9>

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities

of ai-generated text detection. *arXiv preprint arXiv:2304.04736*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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. <https://doi.org/10.18653/v1/N19-1423>

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3558–3567. Association for Computational Linguistics. <https://doi.org/10.18653/v1/p19-1346>

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: BERT-based adversarial exam-

ples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181. <https://doi.org/10.18653/v1/2020.emnlp-main.498>

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved problems in ML safety. *arXiv preprint arXiv:2109.13916*.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. 2018. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*.

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

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. <https://doi.org/10.18653/v1/N18-1170>

Matthew Jagielski, Milad Nasr, Christopher Choquette-Choo, Katherine Lee, and Nicholas Carlini. 2023. Students parrot their teachers: Membership inference on model distillation. *arXiv preprint arXiv:2303.03446*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*. <https://doi.org/10.1609/aaai.v34i05.6311>

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

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. *arXiv preprint arXiv:2303.13408*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202. <https://doi.org/10.18653/v1/2020.emnlp-main.500>

Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and William B. Dolan. 2022. A token-level reference-free hallucination detection benchmark for free-form text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6723–6737.

John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. <https://doi.org/10.48550/arXiv.2301.11305>

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don’t give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745. <https://doi.org/10.18653/v1/D18-1206>

OpenAI. 2023a. AI text classifier.

OpenAI. 2023b. Chatgpt. <https://openai.com/blog/chatgpt/>. Accessed on May 3, 2023.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448. <https://doi.org/10.18653/v1/2022.emnlp-main.225>

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097. <https://doi.org/10.18653/v1/P19-1103>

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 856–865. <https://doi.org/10.18653/v1/P18-1079>

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can AI-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Zhouxing Shi and Minlie Huang. 2020. Robustness to modification with shared words in paraphrase identification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 164–171, Online. <https://doi.org/10.18653/v1/2020.findings-emnlp.16>

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. *arXiv preprint arXiv:1908.09203*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman

Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162. <https://doi.org/10.18653/v1/D19-1221>

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5531–5546. <https://doi.org/10.18653/v1/2020.emnlp-main.446>

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on nlp models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150. <https://doi.org/10.18653/v1/2021.naacl-main.13>

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the robustness of language encoders against grammatical errors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3386–3403, Online. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.acl-main.310>

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

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023. Protecting language generation models via invisible watermarking. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42187–42199. PMLR.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404. <https://doi.org/10.18653/v1/2021.findings-acl.120>

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.