DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature

Eric Mitchell [1] **Yoonho Lee** [1] **Alexander Khazatsky** [1] **Christopher D. Manning** [1] **Chelsea Finn** [1]

Abstract

The increasing fluency and widespread usage of large language models (LLMs) highlight the desirability of corresponding tools aiding detection of LLM-generated text. In this paper, we identify a property of the structure of an LLM's probability function that is useful for such detection. Specifically, we demonstrate that text sampled from an LLM tends to occupy negative curvature regions of the model's log probability function. Leveraging this observation, we then define a new curvature-based criterion for judging if a passage is generated from a given LLM. This approach, which we call DetectGPT, does not require training a separate classifier, collecting a dataset of real or generated passages, or explicitly watermarking generated text. It uses only log probabilities computed by the model of interest and random perturbations of the passage from another generic pre-trained language model (e.g., T5). We find DetectGPT is more discriminative than existing zero-shot methods for model sample detection, notably improving detection of fake news articles generated by 20B parameter GPT-NeoX from 0.81 AUROC for the strongest zero-shot baseline to 0.95 AUROC for DetectGPT. See `ericmitchell.ai/detectgpt` for code, data, and other project information.

1. Introduction

Large language models (LLMs) have proven able to generate remarkably fluent responses to a wide variety of user queries. Models such as GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and ChatGPT (OpenAI, 2022) can convincingly answer complex questions about science, mathematics, historical and current events, and social trends.

[1]Stanford University. Correspondence to: Eric Mitchell <eric.mitchell@cs.stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

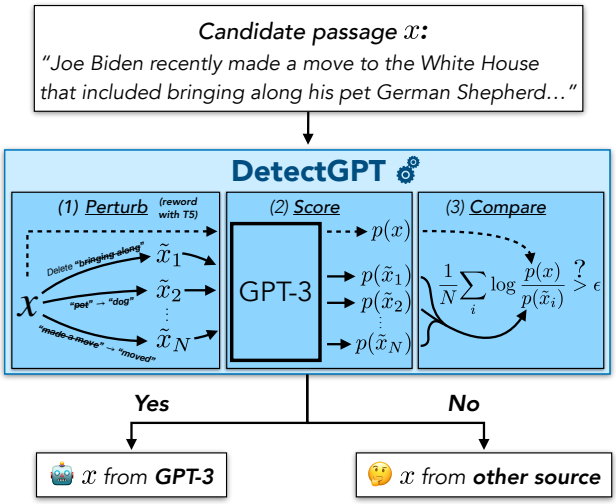

Figure 1. We aim to determine whether a piece of text was generated by a particular LLM p, such as GPT-3. To classify a candidate passage x, DetectGPT first generates minor **perturbations** of the passage $\tilde{x}_i$ using a generic pre-trained model such as T5. Then DetectGPT **compares** the log probability under p of the original sample x with each perturbed sample $\tilde{x}_i$. If the average log ratio is high, the sample is likely from the source model.

While recent work has found that cogent-sounding LLM-generated responses are often simply wrong (Lin et al., 2022), the articulate nature of such generated text may still make LLMs attractive for replacing human labor in some contexts, notably student essay writing and journalism. At least one major news source has released AI-written content with limited human review, leading to substantial factual errors in some articles (Christian, 2023). Such applications of LLMs are problematic for a variety of reasons, making fair student assessment difficult, impairing student learning, and proliferating convincing-but-inaccurate news articles. Unfortunately, humans perform only slightly better than chance when classifying machine-generated vs human-written text (Gehrmann et al., 2019), leading researchers to consider automated detection methods that may identify signals difficult for humans to recognize. Such methods might give teachers and news-readers more confidence in the human origin of the text that they consume.

As in prior work (Jawahar et al., 2020), we study the

machine-generated text detection problem as a binary classification problem. Specifically, we aim to classify whether a *candidate passage* was generated by a particular *source model*. While several works have investigated methods for training a second deep network to detect machine-generated text, such an approach has several shortcomings, including a tendency to overfit to the topics it was trained on as well as the need to train a new model for each new source model that is released. We therefore consider the *zero-shot* version of machine-generated text detection, where we use the source model itself, without fine-tuning or adaptation of any kind, to detect its own samples. The most common method for zero-shot machine-generated text detection is evaluating the average per-token log probability of the generated text and thresholding (Solaiman et al., 2019; Gehrmann et al., 2019; Ippolito et al., 2020). However, this zeroth-order approach to detection ignores the local structure of the learned probability function around a candidate passage, which we find contains useful information about the source of a passage.

This paper poses a simple hypothesis: minor rewrites of *model-generated* text tend to have lower log probability under the model than the original sample, while minor rewrites of *human-written* text may have higher or lower log probability than the original sample. In other words, unlike human-written text, model-generated text tends to lie in areas where the log probability function has negative curvature (e.g., local maxima of the log probability). We empirically verify this hypothesis, and find that it holds true across a diverse body of LLMs, even when the minor rewrites, or *perturbations*, come from alternative language models. We leverage this observation to build DetectGPT, a zero-shot method for automated machine-generated text detection. To test if a passage came from a source model p_θ, DetectGPT compares the log probability of the candidate passage under p_θ with the average log probability of several perturbations of the passage under p_θ (generated with, e.g., T5; Raffel et al. (2020)). If the perturbed passages tend to have lower average log probability than the original by some margin, the candidate passage is likely to have come from p_θ. See Figure 1 for an overview of the problem and DetectGPT. See Figure 2 for an illustration of the underlying hypothesis and Figure 3 for empirical evaluation of the hypothesis. Our experiments find that DetectGPT is more accurate than existing zero-shot methods for detecting machine-generated text, improving over the strongest zero-shot baseline by over 0.1 AUROC for multiple source models when detecting machine-generated news articles.

Contributions. Our main contributions are: (a) the identification and empirical validation of the hypothesis that the curvature of a model's log probability function tends to be significantly more negative at model samples than for human text, and (b) DetectGPT, a practical algorithm inspired by this hypothesis that approximates the trace of the log probability function's Hessian to detect a model's samples.

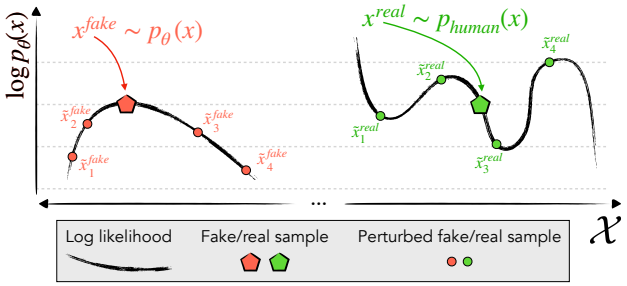

Figure 2. We identify and exploit the tendency of machine-generated passages $x \sim p_\theta(\cdot)$ **(left)** to lie in negative curvature regions of $\log p(x)$, where nearby samples have lower model log probability on average. In contrast, human-written text $x \sim p_{real}(\cdot)$ **(right)** tends not to occupy regions with clear negative log probability curvature.

2. Related Work

Increasingly large LLMs (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2022; Zhang et al., 2022) have led to dramatically improved performance on many language-related benchmarks and the ability to generate convincing and on-topic text. GROVER (Zellers et al., 2019) was the first LLM trained specifically for generating plausible news articles. Human evaluators found GROVER-generated propaganda at least as trustworthy as human-written propaganda, motivating the authors to study GROVER's ability to detect its own generations by fine-tuning a detector on top of its features; they found GROVER better able to detect GROVER-generated text than other pre-trained models. However, Bakhtin et al. (2019); Uchendu et al. (2020) note that models trained explicitly to detect machine-generated text tend to overfit to their training distribution of data or source models.

Other works have trained supervised models for machine-generated text detection on top of neural representations (Bakhtin et al., 2019; Solaiman et al., 2019; Uchendu et al., 2020; Ippolito et al., 2020; Fagni et al., 2021), bag-of-words features (Solaiman et al., 2019; Fagni et al., 2021), and hand-crafted statistical features (Gehrmann et al., 2019). Alternatively, Solaiman et al. (2019) notes the surprising efficacy of a simple zero-shot method for machine-generated text detection, which thresholds a candidate passage based on its average log probability under the generative model, serving as a strong baseline for zero-shot machine-generated text detection in our work. In our work, we similarly use the generating model to detect its own generations in a zero shot manner, but through a different approach based on estimating local curvature of the log probability around the sample rather than the raw log probability of the sample itself. See Jawahar et al. (2020) for a complete survey on machine-

generated text detection. Other work explores watermarks for generated text (Kirchenbauer et al., 2023), which modify a model's generations to make them easier to detect. Our work does not assume text is generated with the goal of easy detection; DetectGPT detects text generated from publicly available LLMs using standard LLM sampling strategies.

The widespread use of LLMs has led to much other contemporaneous work on detecting LLM output. Sadasivan et al. (2023) show that the detection AUROC of the an detector is upper bounded by a function of the TV distance between the model and human text. However, we find that AUROC of DetectGPT is high even for the largest publicly-available models (Table 2), suggesting that TV distance may not correlate strongly with model scale and capability. This disconnect may be exacerbated by new training objectives other than maximum likelihood, notably reinforcement learning with human feedback (Christiano et al., 2017; Ziegler et al., 2020). Both Sadasivan et al. (2023) and Krishna et al. (2023) show the effectiveness of paraphrasing as a tool for evading detection, suggesting an important area of study for future work. Liang et al. (2023) show that multilingual detection is difficult, with non-DetectGPT detectors showing bias against non-native speakers; this result highlights the advantage of zero-shot detectors like DetectGPT, which generalize well to any data generated by the original generating model. Finally, Mireshghallah et al. (2023) perform a comprehensive study of which proxy scoring models produce the most useful log probabilities for detection when the generating model is not known (a large-scale version of our Figure 6). Surprisingly (but consistent with our findings), they find that smaller models are in fact *better* proxy models for performing detection with perturbation-based methods like DetectGPT.

The problem of machine-generated text detection echoes earlier work on detecting deepfakes, artificial images or videos generated by deep nets, which has spawned substantial efforts in detection of fake visual content (Dolhansky et al., 2020; Zi et al., 2020). While early works in deepfake detection used relatively general-purpose model architectures (Güera & Delp, 2018), many deepfake detection methods rely on the continuous nature of image data to achieve state-of-the-art performance (Zhao et al., 2021; Guarnera et al., 2020), making direct application to text difficult.

3. The Zero-Shot Machine-Generated Text Detection Problem

We study zero-shot machine-generated text detection, the problem of detecting whether a piece of text, or *candidate passage* x, is a sample from a *source model* p_θ. The problem is zero-shot in the sense that we do not assume access to human-written or generated samples to perform detection. As in prior work, we study a 'white box' setting (Gehrmann et al., 2019) in which the detector may evaluate the log probability of a sample $\log p_\theta(x)$. The white box setting **does not** assume access to the model architecture or parameters. Most public APIs for LLMs (such as GPT-3) enable scoring text, though some exceptions exist, notably ChatGPT. While most of our experiments consider the white box setting, see Section 5.2 for a suite of experiments in which we score text using models other than the source model. See Mireshghallah et al. (2023) for a comprehensive evaluation in this setting.

Algorithm 1 DetectGPT model-generated text detection

Input: passage x, source model p_θ, perturbation function q, number of perturbations k, decision threshold ϵ
$\tilde{x}_i \sim q(\cdot \mid x)$, $i \in [1..k]$ // mask spans, sample replacements
$\tilde{\mu} \leftarrow \frac{1}{k}\sum_i \log p_\theta(\tilde{x}_i)$ // approximate expectation in Eq. 1
$\hat{\mathbf{d}}_x \leftarrow \log p_\theta(x) - \tilde{\mu}$ // estimate $\mathbf{d}(x, p_\theta, q)$
$\tilde{\sigma}_x^2 \leftarrow \frac{1}{k-1}\sum_i (\log p_\theta(\tilde{x}_i) - \tilde{\mu})^2$ // variance for normalization
if $\frac{\hat{\mathbf{d}}_x}{\sqrt{\tilde{\sigma}_x}} > \epsilon$ **then**
  return `true` // probably model sample
else
  return `false` // probably not model sample

The detection criterion we propose, DetectGPT, also makes use of generic pre-trained mask-filling models in order to generate passages that are 'nearby' the candidate passage. However, these mask-filling models are used off-the-shelf, without any fine-tuning or adaptation to the target domain.

4. DetectGPT: Zero-shot Machine-Generated Text Detection with Random Perturbations

DetectGPT is based on the hypothesis that samples from a source model p_θ typically lie in areas of negative curvature of the log probability function of p_θ, unlike human text. In other words, if we apply small perturbations to a passage $x \sim p_\theta$, producing $\tilde{x}$, the quantity $\log p_\theta(x) - \log p_\theta(\tilde{x})$ should be relatively large on average for machine-generated samples compared to human-written text. To leverage this hypothesis, first consider a perturbation function $q(\cdot \mid x)$ that gives a distribution over $\tilde{x}$, slightly modified versions of x with similar meaning (we will generally consider roughly paragraph-length texts x). As an example, $q(\cdot \mid x)$ might be the result of simply asking a human to rewrite one of the sentences of x, while preserving the meaning of x. Using the notion of a perturbation function, we can define the *perturbation discrepancy* $\mathbf{d}(x, p_\theta, q)$:

$$\mathbf{d}(x, p_\theta, q) \triangleq \log p_\theta(x) - \mathbb{E}_{\tilde{x}\sim q(\cdot|x)} \log p_\theta(\tilde{x}) \quad (1)$$

We state our hypothesis more formally as the Local Perturbation Discrepancy Gap Hypothesis, which describes a gap in the perturbation discrepancy for model-generated text and human-generated text.

Perturbation Discrepancy Gap Hypothesis. *If q produces samples on the data manifold, $\mathbf{d}(x, p_\theta, q)$ is positive with*

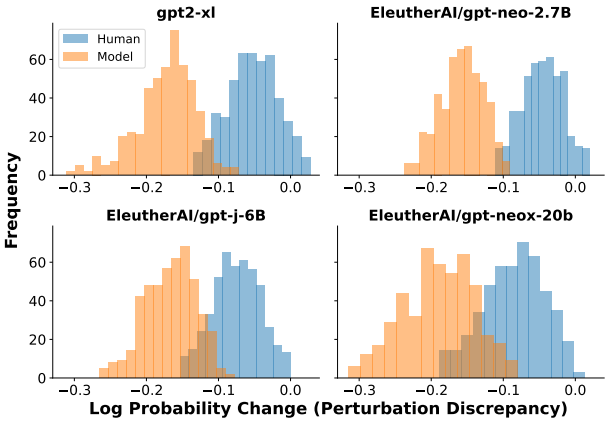

Figure 3. The average drop in log probability (perturbation discrepancy) after rephrasing a passage is consistently higher for model-generated passages than for human-written passages. Each plot shows the distribution of the perturbation discrepancy $\mathbf{d}(x, p_\theta, q)$ for **human-written news articles** and **machine-generated articles** of equal word length. Human-written articles are a sample of 500 XSum articles; machine-generated text, generated from models GPT-2 (1.5B), GPT-Neo-2.7B (Black et al., 2021), GPT-J (6B; Wang & Komatsuzaki (2021)) and GPT-NeoX (20B; Black et al. (2022)), is generated by prompting each model with the first 30 tokens of each XSum article, sampling from the raw conditional distribution. Discrepancies are estimated with 100 T5-3B samples.

high probability for samples $x \sim p_\theta$. For human-written text, $\mathbf{d}(x, p_\theta, q)$ tends toward zero for all x.

If we define $q(\cdot \mid x)$ to be samples from a mask-filling model such as T5 (Raffel et al., 2020), rather than human rewrites, we can empirically test the Perturbation Discrepancy Gap Hypothesis in an automated, scalable manner. For real data, we use 500 news articles from the XSum dataset (Narayan et al., 2018); for model samples, we use the output of four different LLMs when prompted with the first 30 tokens of each article in XSum. We use T5-3B to apply perturbations, masking out randomly-sampled 2-word spans until 15% of the words in the article are masked. We approximate the expectation in Eq. 1 with 100 samples from T5.[1] Figure 3 shows the result of this experiment. We find the distribution of perturbation discrepancies is significantly different for human-written articles and model samples; model samples tend to have a larger perturbation discrepancy. Section 5.3 explores a relaxation of the assumption that q only produces samples on the data manifold, finding that a gap, although reduced, still exists in this case.

Given these results, we can detect if a piece of text was generated by a model p_θ by simply thresholding the perturbation discrepancy. In practice, we find that normalizing the perturbation discrepancy by the standard deviation of the observed values used to estimate $\mathbb{E}_{\tilde{x} \sim q(\cdot|x)} \log p_\theta(\tilde{x})$ provides a slightly better signal for detection, typically increasing AUROC by around 0.020, so we use this normalized version of the perturbation discrepancy in our experiments. The resulting method, DetectGPT, is summarized in Alg. 1. Having described an application of the perturbation discrepancy to machine-generated text detection, we next provide an interpretation of this quantity.

Interpretation of perturbation discrepancy as curvature

While Figure 3 suggests that the perturbation discrepancy may be useful, it is not immediately obvious what it measures. In this section, we show that the perturbation discrepancy approximates a measure of the local curvature of the log probability function near the candidate passage, more specifically, that it is proportional to the negative trace of the Hessian of the log probability function.[2] To handle the non-differentiability of discrete data, we consider candidate passages in a latent semantic space, where small displacements correspond to valid edits that retain similar meaning to the original. Because our perturbation function (T5) models natural text, we expect our perturbations to roughly capture such meaningful variations of the original passage, rather than arbitrary edits.

We first invoke Hutchinson's trace estimator (Hutchinson, 1990), giving an unbiased estimate of the trace of matrix A:

$$\operatorname{tr}(A) = \mathbb{E}_{\mathbf{z}} \mathbf{z}^\top A \mathbf{z} \tag{2}$$

provided that the elements of $\mathbf{z} \sim q_z$ are IID with $\mathbb{E}[z_i] = 0$ and $\operatorname{Var}(z_i) = 1$. To use Equation 2 to estimate the trace of the Hessian, we must therefore compute the expectation of the directional second derivative $\mathbf{z}^\top H_f(x)\mathbf{z}$. We approximate this expression with finite differences:

$$\mathbf{z}^\top H_f(x)\mathbf{z} \approx \frac{f(x + h\mathbf{z}) + f(x - h\mathbf{z}) - 2f(x)}{h^2} \tag{3}$$

Combining Equations 2 and 3 and simplifying with $h = 1$, we have an estimate of the negative Hessian trace

$$-\operatorname{tr}(H)_f(x) \approx 2f(x) - \mathbb{E}_{\mathbf{z}}\left[f(x + \mathbf{z}) + f(x - \mathbf{z})\right]. \tag{4}$$

If our noise distribution is *symmetric*, that is, $p(\mathbf{z}) = p(-\mathbf{z})$ for all $\mathbf{z}$, then we can simplify Equation 4 to

$$\frac{-\operatorname{tr}(H)_f(x)}{2} \approx f(x) - \mathbb{E}_{\mathbf{z}} f(x + \mathbf{z}). \tag{5}$$

We note that the RHS of Equation 5 corresponds to the perturbation discrepancy (1) where the perturbation function $q(\tilde{x} \mid x)$ is replaced by the distribution $q_z(z)$ used

[1] We later show in Figure 8 that varying the number of samples used to estimate the expectation effectively allows for trading off between accuracy and speed.

[2] Rather than the Hessian of the log likelihood with respect to model parameters (the Fisher Information Matrix), here we refer to the Hessian of the log probability with respect to the sample x.

	XSum						SQuAD						WritingPrompts					
Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**
$\log p(x)$	0.86	0.86	0.86	0.82	0.77	0.83	0.91	0.88	0.84	0.78	0.71	0.82	0.97	0.95	0.95	0.94	0.93*	0.95
Rank	0.79	0.76	0.77	0.75	0.73	0.76	0.83	0.82	0.80	0.79	0.74	0.80	0.87	0.83	0.82	0.83	0.81	0.83
LogRank	0.89*	0.88*	0.90*	0.86*	0.81*	0.87*	0.94*	0.92*	0.90*	0.83*	0.76*	0.87*	0.98*	0.96*	0.97*	0.96*	**0.95**	0.96*
Entropy	0.60	0.50	0.58	0.58	0.61	0.57	0.58	0.53	0.58	0.58	0.59	0.57	0.37	0.42	0.34	0.36	0.39	0.38
DetectGPT	**0.99**	**0.97**	**0.99**	**0.97**	**0.95**	**0.97**	**0.99**	**0.97**	**0.97**	**0.90**	**0.79**	**0.92**	**0.99**	**0.99**	**0.99**	**0.97**	0.93*	**0.97**
Diff	0.10	0.09	0.09	0.11	0.14	0.10	0.05	0.05	0.07	0.07	0.03	0.05	0.01	0.03	0.02	0.01	-0.02	0.01

Table 1. AUROC for detecting samples from the given model on the given dataset for DetectGPT and four previously proposed criteria (500 samples used for evaluation). From 1.5B parameter GPT-2 to 20B parameter GPT-NeoX, DetectGPT consistently provides the most accurate detections. **Bold** shows the best AUROC within each column (model-dataset combination); asterisk (*) denotes the second-best AUROC. Values in the final row show DetectGPT's AUROC over the strongest baseline method in that column.

in Hutchinson's trace estimator (2). Here, $\tilde{x}$ is a high-dimensional sequence of tokens while q_z is a vector in a compact semantic space. Since the mask-filling model samples sentences similar to x with minimal changes to semantic meaning, we can think of the mask-filling model as first sampling a similar semantic embedding ($\tilde{z} \sim q_z$) and then mapping this to a token sequence ($\tilde{z} \mapsto \tilde{x}$). Sampling in semantic space ensures that all samples stay near the data manifold, which is useful because we would expect the log probability to always drop if we randomly perturb tokens. We can therefore interpret our objective as approximating the curvature restricted to the data manifold.

5. Experiments

We conduct experiments to better understand multiple facets of machine-generated text detection; we study the effectiveness of DetectGPT for zero-shot machine-generated text detection compared to prior zero-shot approaches, the impact of distribution shift on zero-shot and supervised detectors, and detection accuracy for the largest publicly-available models. To further characterize factors that impact detection accuracy, we also study the robustness of zero-shot methods to machine-generated text that has been partially revised, the impact of alternative decoding strategies on detection accuracy, and a black-box variant of the detection task. Finally, we analyze more closely DetectGPT's behavior as the choice of perturbation function, the number of samples used to estimate $\mathbf{d}(x, p_\theta, q)$, the length of the passage, and the data distribution is varied.

Comparisons. We compare DetectGPT with various existing zero-shot methods for machine-generated text detection that also leverage the predicted token-wise conditional distributions of the source model for detection. These methods correspond to statistical tests based on token log probabilities, token ranks, or predictive entropy (Gehrmann et al., 2019; Solaiman et al., 2019; Ippolito et al., 2020). The first method uses the source model's average token-wise log probability to determine if a candidate passage is machine-generated or not; passages with high average log probability are likely to be generated by the model. The second and third methods use the average observed rank or log-rank of the tokens in the candidate passage according to the model's conditional distributions. Passages with smaller average (log-)rank are likely machine-generated. We also evaluate an entropy-based approach inspired by the hypothesis in Gehrmann et al. (2019) that model-generated texts will be more 'in-distribution' for the model, leading to more over-confident (thus lower entropy) predictive distributions. Empirically, we find predictive entropy to be *positively* correlated with passage fake-ness more often that not; therefore, this baseline uses high average entropy in the model's predictive distribution as a signal that a passage is machine-generated. While our main focus is on zero-shot detectors as they do not require re-training for new domains or source models, for completeness we perform comparisons to supervised detection models in Section 5.1, using OpenAI's RoBERTa-based (Liu et al., 2019) GPT-2 detector models,[3] which are fine-tuned on millions of samples from various GPT-2 model sizes and decoding strategies.

Datasets & metrics Our experiments use six datasets that cover a variety of everyday domains and LLM use-cases. We use news articles from the XSum dataset (Narayan et al., 2018) to represent fake news detection, Wikipedia paragraphs from SQuAD contexts (Rajpurkar et al., 2016) to represent machine-written academic essays, and prompted stories from the Reddit WritingPrompts dataset (Fan et al., 2018) to represent detecting machine-generated creative writing submissions. To evaluate robustness to distribution shift, we also use the English and German splits of WMT16 (Bojar et al., 2016) as well as long-form answers written by human experts in the PubMedQA dataset (Jin et al., 2019). Each experiment uses between 150 and 500 examples for evaluation, as noted in the text. For each experiment, we generate the machine-generated text by prompting with the first 30 tokens of the real text (or just the question tokens for the PubMedQA experiments). We measure performance using the area under the receiver operating characteristic curve (AUROC), which can be interpreted as the probability that a classifier correctly ranks a randomly-selected positive (machine-generated) example higher than a randomly-

[3] https://github.com/openai/gpt-2-output-dataset/tree/master/detector

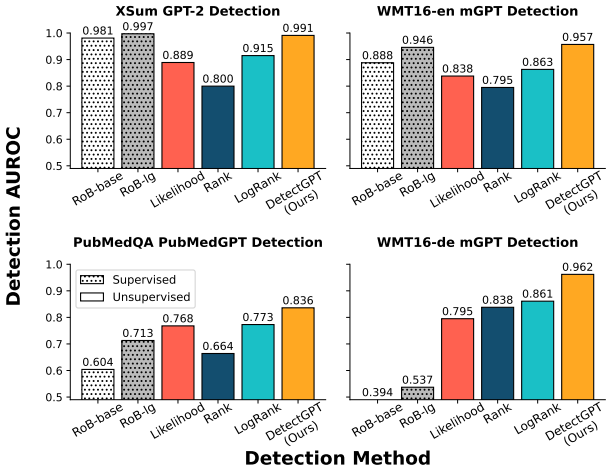

Figure 4. Supervised machine-generated text detection models trained on large datasets of real and generated texts perform as well as or better than DetectGPT on **in-distribution (top row)** text. However, zero-shot methods work out-of-the-box for **new domains (bottom row)** such as PubMed medical texts and German news data from WMT16. For these domains, supervised detectors fail due to excessive distribution shift.

selected negative (human-written) example. All experiments use an equal number of positive and negative examples.

Hyperparameters. The key hyperparameters of DetectGPT are the fraction of words masked for perturbation, the length of the masked spans, the model used for mask filling, and the sampling hyperparameters for the mask-filling model. Using BERT (Devlin et al., 2019) masked language modeling as inspiration, we use 15% as the mask rate. We performed a small sweep over masked span lengths of $\{2, 5, 10\}$ on a held-out set of XSum data, finding 2 to perform best. We use these settings for **all experiments, without re-tuning**. We use T5-3B for almost all experiments, except for GPT-NeoX and GPT-3 experiments, where compute resources allowed for the larger T5-11B model; we also use mT5-3B instead of T5-3B for the WMT multilingual experiment. We do not tune the hyperparameters for the mask filling model, sampling directly with temperature 1.

5.1. Main Results

We first present two groups of experiments to evaluate DetectGPT along with existing methods for zero-shot and supervised detection on models from 1.5B to 175B parameters.

Zero-shot machine-generated text detection. We present the comparison of different zero-shot detection methods in Table 1. In these experiments, model samples are generated by sampling from the raw conditional distribution with temperature 1. DetectGPT most improves average detection accuracy for XSum stories (0.1 AUROC improvement) and SQuAD Wikipedia contexts (0.05 AUROC improvement). While it also performs accurate detection for Writing-

	PMQA	XSum	WritingP	Avg.
RoB-base	0.64 / 0.58	**0.92** / 0.74	**0.92** / 0.81	0.77
RoB-large	0.71 / 0.64	**0.92** / **0.88**	0.91 / **0.88**	**0.82**
$\log p(x)$	0.64 / 0.55	0.76 / 0.61	0.88 / 0.67	0.69
DetectGPT	**0.84** / **0.77**	0.84 / 0.84	0.87 / 0.84	**0.83**

Table 2. DetectGPT detects generations from GPT-3 and Jurassic-2 Jumbo (175B models from OpenAI and AI21 Labs) with average AUROC on-par with supervised models trained specifically for machine-generated text detection. For more 'typical' text, such as news articles, supervised methods perform strongly. The GPT-3 AUROC appears first in each column, the Jurassic-2 AUROC appears second (i.e., after the slash).

Prompts, the performance of all methods tends to increase, and the average margin of improvement is narrow.[4] For 14 of the 15 combinations of dataset and model, DetectGPT provides the most accurate detection performance, with a 0.06 AUROC improvement on average. Log-rank thresholding proves to be a consistently stronger baseline than log probability thresholding, although it requires slightly more information (full predicted logits), which are not always available in public APIs.

Comparison with supervised detectors. While our experiments generally focus on zero-shot detection, some works have evaluated the detection performance of supervised methods (typically fine-tuned transformers) for detecting machine-generated text. In this section, we explore several domains to better understand the relative strengths of supervised and zero-shot detectors. The results are presented in Figure 4, using 200 samples from each dataset for evaluation. We find that supervised detectors can provide similar detection performance to DetectGPT on *in-distribution* data like English news, but perform significantly worse than zero-shot methods in the case of English scientific writing and fail altogether for German writing. This finding echoes past work showing that language models trained for machine-generated text detection overfit to their training data (source model, decoding strategy, topic, language, etc.; Uchendu et al. (2020); Ippolito et al. (2020); Jawahar et al. (2020)). In contrast, zero-shot methods generalize relatively easily to new languages and domains; DetectGPT's performance in particular is mostly unaffected by the change in language from English to German.

While our experiments have shown that DetectGPT is effective on a variety of domains and models, it is natural to wonder if it is effective for the largest publicly-available LMs. Therefore, we also evaluate multiple zero-shot and supervised methods on two 175B parameter models, OpenAI's GPT-3 and AI21 Labs' Jurassic-2 Jumbo. Because neither

[4]The overall ease of detecting machine-generated fake writing corroborates anecdotal reporting that machine-generated creative writing tends to be noticeably generic, and therefore relatively easy to detect (Roose & Newton, 2022).

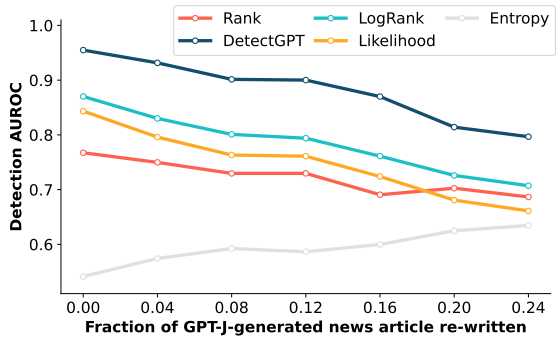

Figure 5. We simulate human edits to machine-generated text by replacing varying fractions of model samples with T5-3B generated text (masking out random five word spans until $r\%$ of text is masked to simulate human edits to machine-generated text). The four top-performing methods all generally degrade in performance with heavier revision, but DetectGPT is consistently most accurate. Experiment is conducted on the XSum dataset.

API provides access to the complete conditional distribution for each token, we cannot compare to the rank, log rank, and entropy-based prior methods. We sample 150 examples[5] from the PubMedQA, XSum, and WritingPrompts datasets and compare the two pre-trained RoBERTa-based detector models with DetectGPT and the probability thresholding baseline. We show in Table 2 that DetectGPT can provide detection competitive with or better than the stronger of the two supervised models, and it again greatly outperforms probability thresholding on average.

5.2. Variants of Machine-Generated Text Detection

Detecting paraphrased machine-generated text. In practice, humans may manually edit or refine machine-generated text rather than blindly use a model's generations for their task of interest. We therefore conduct an experiment to simulate the detection problem for model samples that have been increasingly heavily revised. We simulate human revision by replacing 5 word spans of the text with samples from T5-3B until $r\%$ of the text has been replaced, and report performance as r varies. Figure 5 shows that DetectGPT maintains detection AUROC above 0.8 even when nearly a quarter of the text in model samples has been replaced. Unsurprisingly, almost all methods show a gradual degradation in performance as the sample is more heavily revised. The entropy baseline shows surprisingly robust performance in this setting (althought it is least accurate on average), even slightly improving detection performance up to 24% replacement. DetectGPT shows the strongest detection performance for all revision levels.

Impact of alternative decoding strategies on detection.

[5]We reduce the number of evaluation samples from 500 in our main experiments to reduce the API costs of these experiments.

	XSum		SQuAD		WritingPrompts	
Method	top-p	top-k	top-p	top-k	top-p	top-k
$\log p(x)$	0.92	0.87	0.89	0.85	**0.98**	0.96
Rank	0.76	0.76	0.81	0.80	0.84	0.83
LogRank	0.93*	0.90*	0.92*	0.90*	**0.98**	**0.97**
Entropy	0.53	0.55	0.54	0.56	0.32	0.35
DetectGPT	**0.98**	**0.98**	**0.94**	**0.93**	**0.98**	**0.97**

Table 3. AUROC for zero-shot methods averaged across the five models in Table 1 for both top-k and top-p sampling, with $k = 40$ and $p = 0.96$. Both settings enable slightly more accurate detection, and DetectGPT consistently provides the best detection performance. See Appendix Tables 4 and 5 for complete results.

While Table 1 suggests that DetectGPT is effective for detecting machine-generated text, prior work notes that the decoding strategy (i.e., temperature sampling, top-k, nucleus/top-p) can impact the difficulty of detection. We repeat the analysis from Section 5.1 using top-k sampling and nucleus sampling. Top-k sampling truncates the sampling distribution to only the k highest-probability next tokens; nucleus sampling samples from only the smallest set of tokens whose combined probability exceeds p. The results are summarized in Table 3; Appendix Tables 4 and 5 show complete results. We use $k = 40$, and $p = 0.96$, in line with prior work (Ippolito et al., 2020). We find that both top-k and nucleus sampling make detection easier, on average. Averaging across domains, DetectGPT provides the clearest signal for zero-shot detection.

Detection when the source model is unknown. While our experiments have focused on the white-box setting for machine-generated text detection, in this section, we explore the effect of using a different model to *score* a candidate passage (and perturbed texts) than the model that generated the passage. In other words, we aim to classify between human-generated text and text from model A, but without access to model A to compute log probabilities. Instead, we use log probabilities computed by a surrogate model B. We consider three models, GPT-J, GPT-Neo-2.7, and GPT-2, evaluating all possible combinations of source model and surrogate model (9 total). We average the performance across 200 samples

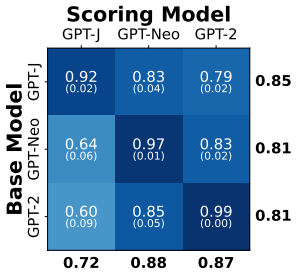

Figure 6. DetectGPT performs best when scoring samples with the same model that generated them (diagonal), but the column means suggest that some models (GPT-Neo, GPT-2) may be better 'scorers' than others (GPT-J). White values show mean (standard error) AUROC over XSum, SQuAD, and WritingPrompts; **black** shows row/column mean.

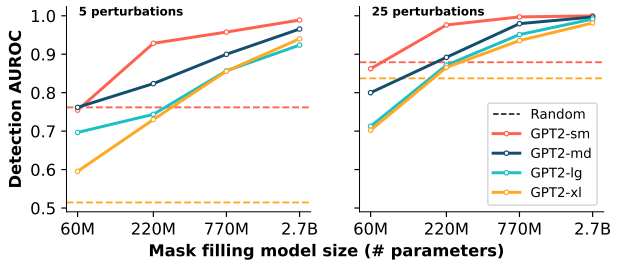

Figure 7. There is a clear association between capacity of mask-filling model and detection performance, across source model scales. Random mask filling (uniform sampling from mask filling model vocabulary) performs poorly, reinforcing the idea that the perturbation function should produce samples on the data manifold. Curves show AUROC scores on 200 SQuAD contexts.

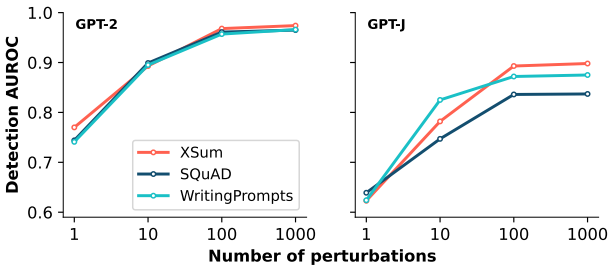

Figure 8. Impact of varying the number of perturbations (samples of mask and mask-fill) used by DetectGPT on AUROC for GPT-2 (**left**) and GPT-J (**right**) to estimate the perturbation discrepancy on detection. Averaging up to 100 perturbations greatly increases DetectGPT's reliability. Perturbations sampled from T5-large.

from XSum, SQuAD, and WritingPrompts. The results are presented in Figure 6, showing that when the surrogate model is different from the source model, detection performance is reduced, indicating that DetectGPT is most suited to the white-box setting. Yet we also observe that if we fix the model used for scoring and average across source models whose generations are detected (average within column), there is significant variation in AUROC; GPT-2 and GPT-Neo-2.7 seem to be better 'scorers' than GPT-J. These variations in cross-model scoring performance suggest ensembling scoring models may be a useful direction for future research; see Mireshghallah et al. (2023) for reference.

5.3. Other factors impacting performance of DetectGPT

In this section, we explore how factors such as the size of the mask-filling model, the number of perturbations used to estimate the expectation in Equation 1, or the data distribution of the text to be detected impact detection quality.

Source and mask-filling model scale. Here we study the impact of the size of the source model and mask-filling model on DetectGPT's performance; the results are shown in Figure 7. In particular, the increased discrimination power of DetectGPT for larger mask-filling models supports the interpretation that DetectGPT is estimating the curvature of the log probability in a latent semantic space, rather than in raw token embedding space. Larger T5 models better represent this latent space, where random directions correspond to meaningful changes in the text.

Number of perturbations for DetectGPT. We evaluate the performance of DetectGPT as a function of the number of perturbations used to estimate the expectation in Equation 1 on three datasets. The results are presented in Figure 8. Detection accuracy continues to improve until 100 perturbations, where it converges. Evaluations use 100 examples from each dataset.

Data distributional properties. We study more closely the impact of the data distribution on DetectGPT, particularly how the domain impacts the threshold separating the perturbation discrepancy distributions of model-generated and human texts as well as the impact of passage length on detection. Figure 9 shows the perturbation discrepancy distributions for model-generated and human texts across four data distributions, using GPT-Neo-2.7B to generate samples. A threshold of slightly below 0.1 separates human and model texts across data distributions, which is important for practical scenarios in which a passage may be analyzed without knowing its domain a priori. Finally, Figure 10 shows an analysis of DetectGPT's performance as a function of passage length. We bin the paired human- and model-generated sequences by their average length into three bins of equal size (bottom/middle/top third), and plot the AUROC within each bin. The relationship between detection performance and passage length generally depends on the dataset and model (or tokenizer). For very long sequences, DetectGPT may see reduced performance because our implementation of DetectGPT applies all T5 mask-filling perturbations at once, and T5 may fail to track many mask tokens at once. By applying perturbations in multiple sequential rounds of smaller numbers of masks, this effect may be mitigated.

6. Discussion

As large language models continue to improve, they will become increasingly attractive tools for replacing human writers in a variety of contexts, such as education, journalism, and art. While legitimate uses of language model technologies exist in all of these settings, teachers, readers, and consumers are likely to demand tools for verifying the human origin of certain content with high educational, societal, or artistic significance, particularly when factuality (and not just fluency) is crucial.

In light of these elevated stakes and the regular emergence of new large language models, we study the *zero-shot machine-generated text detection* problem, in which we use only the raw log probabilities computed by a generative model to determine if a candidate passage was sampled from it. We

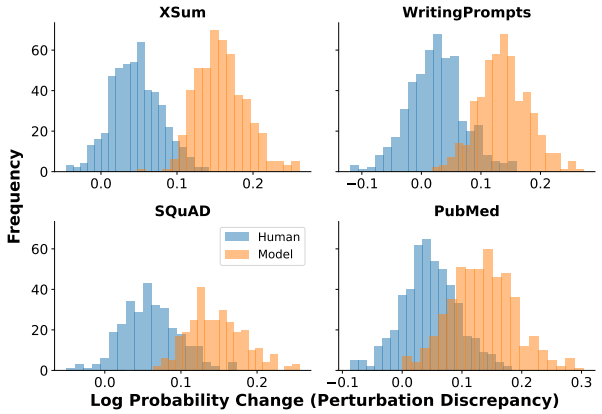

Figure 9. Perturbation discrepancy distributions for GPT-Neo (2.7B) and humans across domains. A threshold of 0.1 generally separates model- and human-generated text well, which is important for practical scenarios where the domain is unknown.

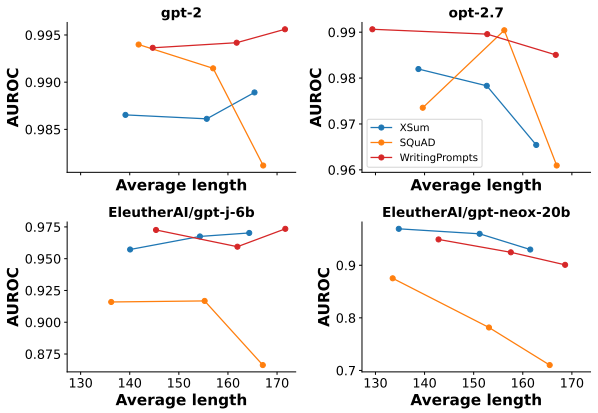

Figure 10. DetectGPT AUROC vs passage length. The relationship between detection performance and passage length generally depends on the dataset and model (or tokenizer). Decreases in detection quality with increasing length may be due to T5 failing to track many (20+) masks to fill at once; this problem may be mitigated by applying mask-fills in a sequence of smaller batches.

identify a property of the log probability function computed by a wide variety of large language models, showing that a tractable approximation to the trace of the Hessian of the model's log probability function provides a useful signal for detecting model samples. Our experiments find that this signal is more discriminative than existing zero-shot detection methods and is competitive with bespoke detection models trained with millions of model samples.

DetectGPT and Watermarking. One interpretation of the perturbation function is producing *semantically similar rephrasings of the original passage*. If these rephrasings are systematically lower-probability than the original passage, the model is exposing its bias toward the specific (and roughly arbitrary, by human standards) phrasing used. In other words, LLMs that do not perfectly imitate human writing essentially watermark themselves implicitly. Under this interpretation, efforts to *manually* add watermarking biases to model outputs (Aaronson, 2022; Kirchenbauer et al., 2023) may further improve the effectiveness of methods such as DetectGPT, even as LLMs continue to improve.

Limitations. One limitation of probability-based methods for zero-shot machine-generated text detection (like DetectGPT) is the white-box assumption that we can evaluate log probabilities of the model(s) in question. For models behind APIs that do provide probabilities (such as GPT-3), evaluating probabilities nonetheless costs money. Another assumption of DetectGPT is access to a reasonable perturbation function. While in this work, we use off-the-shelf mask-filling models such as T5 and mT5 (for non-English languages), some domains may see reduced performance if existing mask-filling models do not well represent the space of meaningful rephrases, reducing the quality of the curvature estimate. While DetectGPT provides the best available detection performance for PubMedQA, its drop in performance compared to other datasets may be a result of lower quality perturbations. Finally, DetectGPT is more compute-intensive than other methods for detection, as it requires sampling and scoring the set of perturbations for each candidate passage, rather than just the candidate passage; a better tuned perturbation function or more efficient curvature approximation may help mitigate these costs.

Future Work. While the methods in this work make no assumptions about the models generating the samples, future work may explore how watermarking algorithms can be used in conjunction with detection algorithms like DetectGPT to further improve detection robustness as language models continually improve their reproductions of human text. Separately, the results in Section 5.2 suggest that extending DetectGPT to use ensembles of models for scoring, rather than a single model, may improve detection in the black box setting. Another topic that remains unexplored is the relationship between prompting and detection; that is, can a clever prompt successfully prevent a model's generations from being detected by existing methods? Finally, future work may explore whether the local log probability curvature property we identify is present for generative models in other domains, such as audio, video, or images. We hope that the present work serves as inspiration to future work developing effective, general-purpose methods for mitigating potential harms of machine-generated media.

Acknowledgements

EM gratefully acknowledges funding from a Knight-Hennessy Graduate Fellowship. CF and CM are CIFAR Fellows. The Stanford Center for Research on Foundation Models (CRFM) provided part of the compute resources used for the experiments in this work.

References

Aaronson, S. My Projects at OpenAI, Nov 2022. URL https://scottaaronson.blog/?p=6823.

Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., and Szlam, A. Real or fake? Learning to discriminate machine from human generated text. *arXiv*, 2019. URL http://arxiv.org/abs/1906.03351.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://arxiv.org/abs/2204.06745.

Bojar, O. r., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pp. 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Christian, J. CNET secretly used AI on articles that didn't disclose that fact, staff say. https://web.archive.org/web/20230124063916/https://futurism.com/cnet-ai-articles-label, 2023. Accessed: 2023-01-25.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., and Ferrer, C. C. The deepfake detection challenge dataset, 2020. URL https://ai.facebook.com/datasets/dfdc/.

Fagni, T., Falchi, F., Gambini, M., Martella, A., and Tesconi, M. Tweepfake: About detecting deepfake tweets. *PLOS ONE*, 16(5):1–16, 05 2021. doi: 10.1371/journal.pone.0251415. URL https://doi.org/10.1371/journal.pone.0251415.

Fan, A., Lewis, M., and Dauphin, Y. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Gehrmann, S., Strobelt, H., and Rush, A. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 111–116, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3019. URL https://aclanthology.org/P19-3019.

Guarnera, L., Giudice, O., and Battiato, S. Deepfake detection by analyzing convolutional traces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

Güera, D. and Delp, E. J. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, 2018. doi: 10.1109/AVSS.2018.8639163.

Hutchinson, M. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 19(2):433–450, 1990. doi: 10.1080/03610919008812866. URL https://doi.org/10.1080/03610919008812866.

Ippolito, D., Duckworth, D., Callison-Burch, C., and Eck, D. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1808–1822, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.164. URL https://www.aclweb.org/anthology/2020.acl-main.164.

Jawahar, G., Abdul-Mageed, M., and Lakshmanan, L. V. S. Automatic detection of machine generated text: A critical survey. In *International Conference on Computational Linguistics*, 2020.

Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. PubMedQA: A dataset for biomedical research question answering. 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)*, pp. 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models, 2023. URL https://arxiv.org/abs/2301.10226.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023.

Liang, W., Yuksekgonul, M., Mao, Y., Wu, E., and Zou, J. Gpt detectors are biased against non-native english writers. *arXiv preprint arXiv:2304.02819*, 2023.

Lin, S., Hilton, J., and Evans, O. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Mireshghallah, F., Mattern, J., Gao, S., Shokri, R., and Berg-Kirkpatrick, T. Smaller language models are better black-box machine-generated text detectors. *arXiv preprint arXiv:2305.09859*, 2023.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

OpenAI. Chatgpt: Optimizing language models for dialogue. http://web.archive.org/web/20230109000707/https://openai.com/blog/chatgpt/, 2022. Accessed: 2023-01-10.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for

Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Roose, K. and Newton, C. ChatGPT transforms a classroom and is 'M3GAN' real? Hard Fork, a New York Times Podcast, 2022. URL https://www.nytimes.com/2023/01/13/podcasts/hard-fork-chatgpt-teachers-gen-z-cameras-m3gan.html.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., and Wang, J. Release strategies and the social impacts of language models, 2019. URL https://arxiv.org/ftp/arxiv/papers/1908/1908.09203.pdf.

Uchendu, A., Le, T., Shu, K., and Lee, D. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8384–8395, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.673. URL https://aclanthology.org/2020.emnlp-main.673.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. Defending against neural fake news. In *Neural Information Processing Systems*, 2019.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

Zhao, H., Zhou, W., Chen, D., Wei, T., Zhang, W., and Yu, N. Multi-attentional deepfake detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2185–2194, 2021.

Zi, B., Chang, M., Chen, J., Ma, X., and Jiang, Y.-G. Wild-deepfake: A challenging real-world dataset for deepfake detection. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 2382–2390, 2020.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences, 2020.

A. Complete Results for Top-p and Top-k Decoding

Tables 4 and 5 contain the complete results for XSum, SQuAD, and WritingPrompts for the five models considered in Table 1. On average, both top-p and top-k sampling seem to make the detection task easier. This result is perhaps intuitive, as both sampling methods strictly increase the average log likelihood of model generations under the model (as they truncate low-probability tokens, albeit with different heuristics). Therefore methods based on probability or rank of tokens should become more discriminative.

	XSum						**SQuAD**						**WritingPrompts**					
Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**
$\log p(x)$	0.93	0.93	0.94	0.91	0.87	0.92	0.96	0.94	0.91	0.87	0.79	0.89	0.99*	0.98*	0.98*	0.97*	0.97*	**0.98**
Rank	0.80	0.77	0.77	0.75	0.73	0.76	0.84	0.82	0.81	0.80	0.75	0.81	0.87	0.84	0.83	0.83	0.81	0.84
LogRank	0.95*	0.94*	0.96*	0.93*	0.89*	0.93*	0.98*	0.96*	0.94*	**0.90**	**0.83**	0.92*	0.99*	0.98*	0.98*	**0.98**	**0.98**	**0.98**
Entropy	0.55	0.46	0.53	0.54	0.58	0.53	0.53	0.50	0.55	0.56	0.57	0.54	0.32	0.37	0.28	0.32	0.32	0.32
DetectGPT	**0.99**	**0.98**	**1.00**	**0.98**	**0.97**	**0.98**	**0.99**	**0.98**	**0.98**	**0.90**	0.82*	**0.94**	**1.00**	**0.99**	**0.99**	0.97*	0.93	**0.98**
Diff	0.04	0.04	0.04	0.05	0.08	0.05	0.01	0.02	0.04	0.00	-0.01	0.02	0.01	0.01	0.01	-0.01	-0.05	0.00

Table 4. Nucleus (top-p) sampling evaluation with $p = 0.96$. AUROC for detecting samples from the given model on the given dataset for DetectGPT and four previously proposed criteria. Nucleus sampling generally makes detection easier for all methods, but DetectGPT still provides the highest average AUROC. For WritingPrompts, however, the LogRank baseline performs as well as DetectGPT.

	XSum						**SQuAD**						**WritingPrompts**					
Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**
$\log p(x)$	0.89	0.89	0.89	0.84	0.81	0.87	0.93	0.90	0.88	0.82	0.74	0.85	0.97	0.95	0.97	0.96	0.95*	0.96
Rank	0.79	0.77	0.77	0.75	0.73	0.76	0.84	0.82	0.80	0.80	0.75	0.80	0.87	0.84	0.83	0.82	0.81	0.83
LogRank	0.92*	0.91*	0.93*	0.89*	0.85*	0.90*	0.96*	0.94*	0.92*	0.87*	0.79*	0.90*	0.98*	0.97*	0.98*	**0.97**	**0.96**	**0.97**
Entropy	0.58	0.49	0.55	0.56	0.59	0.55	0.55	0.52	0.56	0.56	0.58	0.56	0.35	0.41	0.30	0.34	0.37	0.35
DetectGPT	**0.99**	**0.97**	**0.99**	**0.96**	**0.96**	**0.98**	**0.99**	**0.98**	**0.98**	**0.89**	**0.80**	**0.93**	**0.99**	**0.98**	**0.99**	**0.97**	0.93	**0.97**
Diff	0.07	0.06	0.06	0.07	0.11	0.08	0.03	0.04	0.06	0.02	0.01	0.03	0.01	0.01	0.01	0.00	-0.03	0.00

Table 5. Top-k sampling evaluation with $k = 40$. DetectGPT generally provides the most accurate performance (highest AUROC), although the gap is narrowed comparing to direct sampling, presumably because top-k generations are more generic.