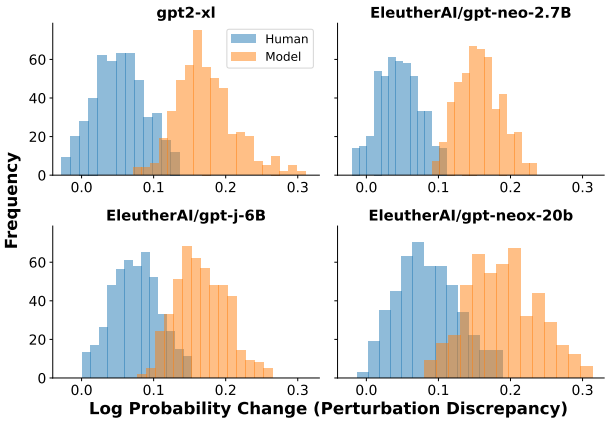

Figure 3. The average drop in log probability (perturbation discrepancy) after rephrasing a passage is consistently higher for model-generated passages than for human-written passages. Each plot shows the distribution of the perturbation discrepancy $\mathbf{d}\,(x, p_\theta, q)$ for **human-written 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.

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 of f at x, 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

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

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

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

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 in Hutchinson's trace estimator (2). Here, $\tilde{x}$ is a high-dimensional sequence of tokens while q_z is a vector in a

[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.

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

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

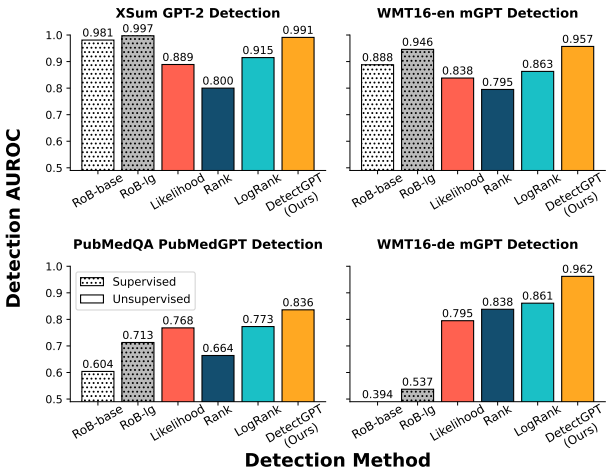

Figure 4. Supervised machine-generated text detection models trained on large datasets of real and generated texts perform as well as or better than DetectGPT on **in-distribution (top row)** text. However, zero-shot methods work out-of-the-box for **new domains (bottom row)** such as PubMed medical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

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 WritingPrompts, the performance of all methods tends to increase,

	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).

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 API provides access to the complete conditional distribution

[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).