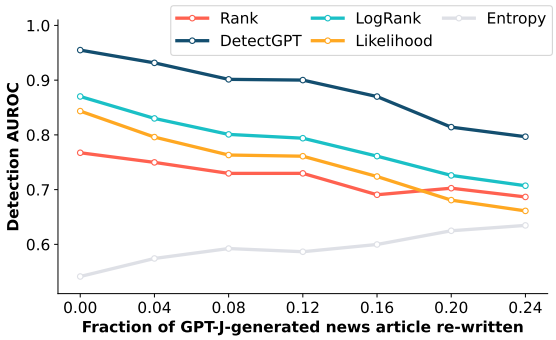

Figure 5. We simulate human edits to machine-generated text by replacing varying fractions of model samples with T5-3B generated text (masking out random five word spans until $r\%$ of text is masked to simulate human edits to machine-generated text). The four top-performing methods all generally degrade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

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. 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.

[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.

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 from XSum, SQuAD, and

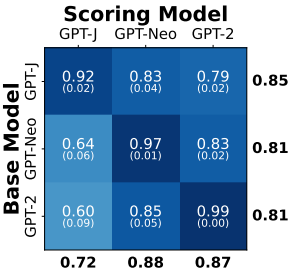

Figure 6. DetectGPT performs best when scoring samples with the same model that generated them (diagonal), but the column means suggest that some models (GPT-Neo, GPT-2) may be better 'scorers' than others (GPT-J). White values show mean (standard error) AUROC over XSum, SQuAD, and WritingPrompts; **black**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

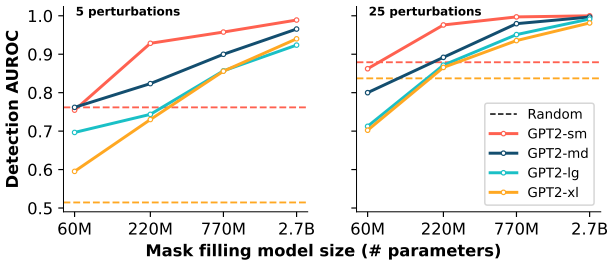

Figure 7. There is a clear association between capacity of mask-filling model and detection performance, across source model scales. Random mask filling (uniform sampling from mask filling model vocabulary) performs poorly, reinforcing the idea that the perturbation function should produce samples on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

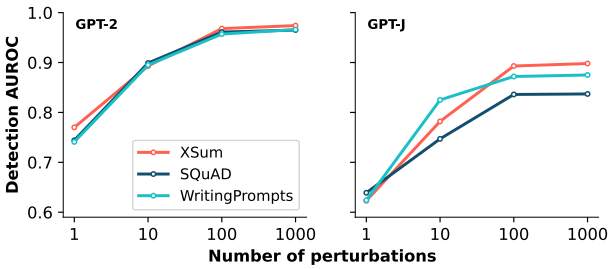

Figure 8. Impact of varying the number of perturbations (samples of mask and mask-fill) used by DetectGPT on AUROC for GPT-2 (**left**) and GPT-J (**right**) to estimate the perturbation discrepancy on detection. Averaging up to 100 perturbations greatly increases DetectGPT's reliability. Perturbations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

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

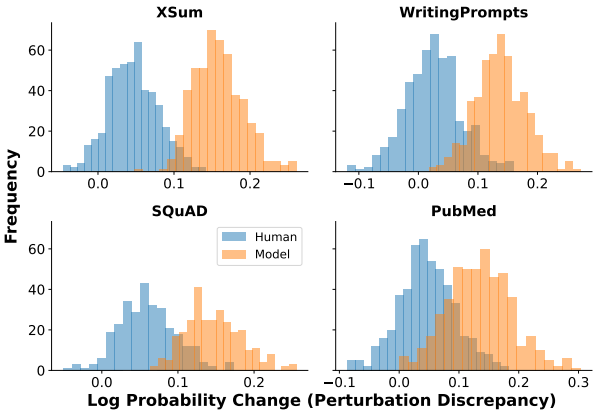

Figure 9. Perturbation discrepancy distributions for GPT-Neo (2.7B) and humans across domains. A threshold of 0.1 generally separates model- and human-generated text well, which is important for practical scenarios where the domain is unknown.

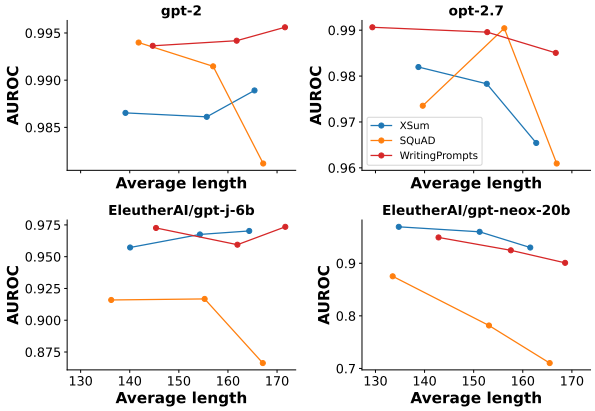

Figure 10. DetectGPT AUROC vs passage length. The relationship between detection performance and passage length generally depends on the dataset and model (or tokenizer). Decreases in detection quality with increasing length may be due to T5 failing to track many (20+) masks to fill at once; this problem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