TempTest: Local Normalization Distortion and the Detection of Machine-generated Text

Tom Kempton[⋆]
Department of Mathematics
University of Manchester

Stuart Burrell[⋆]
Innovation Lab
Featurespace

Connor Cheverall[†]
Institute of Astronomy
University of Cambridge

Abstract

Existing methods for the zero-shot detection of machine-generated text are dominated by three statistical quantities: log-likelihood, log-rank, and entropy. As language models mimic the distribution of human text ever closer, this will limit our ability to build effective detection algorithms. To combat this, we introduce a method for detecting machine-generated text that is entirely agnostic of the generating language model. This is achieved by targeting a defect in the way that decoding strategies, such as temperature or top-k sampling, normalize conditional probability measures. This method can be rigorously theoretically justified, is easily explainable, and is conceptually distinct from existing methods for detecting machine-generated text. We evaluate our detector in the white and black box settings across various language models, datasets, and passage lengths. We also study the effect of paraphrasing attacks on our detector and the extent to which it is biased against non-native speakers. In each of these settings, the performance of our test is at least comparable to that of other state-of-the-art text detectors, and in some cases, we strongly outperform these baselines.

1 INTRODUCTION

There has been a surge of interest in statistical methods for the detection of machine-generated text since the paradigm shift in language modeling triggered by the transformer architecture (Vaswani et al., 2017). This task has growing societal importance, for example, in combating fake news (Ahmed et al., 2023) or ensuring academic integrity (Elkhatat, 2023).

Approaches based on supervised learning typically struggle with out-of-domain generalization (Bao et al., 2024), leading research to focus on the *zero-shot* regime. To date, state-of-the-art zero-shot techniques are typically based on three fundamental statistical quantities: log-likelihood, log-rank, and entropy (Gehrmann et al., 2019). Although state-of-the-art methods such as DetectGPT (Mitchell et al., 2023), Fast-DetectGPT (Bao et al., 2024), DNA-GPT (Yang et al., 2024), and Detect-LLM (Su et al., 2023) have made remarkable progress using these measures, our work demonstrates that a wider suite of basic statistical quantities is beneficial.

We introduce a new zero-shot detection method, TempTest, based on a novel statistical quantity, TempNorm. This statistic is derived from the distortion of conditional probability measures caused by the way that temperature sampling renormalizes probability mass. Temperature sampling is a ubiquitous decoding strategy that allows fine control over the diversity or creativity of generated text. Similar techniques extend to top-k and nucleus (top-p) sampling, although the signal for top-p is very small; see Appendices 9.2 and 9.5.

By explicitly tying detection to artifacts of the text generation process, we are agnostic to the large language model used for generation. This may be important for modern large language models such as Llama 3.1 (Dubey et al., 2024), which in Section 5 we demonstrate cause challenges to existing detectors but not TempTest. In addition, TempTest is easily explainable and rigorously mathematically justified, which are important features for societal adoption given the increasing regulation around the deployment and monitoring of AI models, such as the EU AI act.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

[⋆]Equal contribution.
[†]Work completed while an intern at Featurespace.

In summary, the contributions of this paper are to:

- Introduce TempTest, a new state-of-the-art zero-shot machine-generated text detection method, based on quantifying local normalization distortion of conditional probability measures.

- Validate the robustness of this method across a range of datasets, language models, hyperparameter choices, and white box and black box settings.

- Derive a rigorous justification of this technique based on Bayesian statistics and ergodic theory, showing that it is intuitive and easily explainable.

We emphasize that while TempTest achieves comparable or superior performance to existing art, we do not see ourselves in competition with other detection methods. Instead, our hope is that broadening the set of tools available to our community with a genuinely new approach opens the possibility for nuanced combinations and even better results.

2 BACKGROUND

Problem Formulation We treat the problem of detection of machine-generated text as a binary classification problem. Given a passage of text $\underline{w} = w_1 \dots w_T$ of length $T \in \mathbb{N}$, one wishes to assign labels of 0 and 1 for human and machine-generated text, respectively, via a classifier $f(\underline{w})$. We work in a zero-shot setting, where f does not have learnable parameters optimized via a training process.

Decoding Strategies Given an auto-regressive language model P with vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = N$ and context $\underline{w} = w_{i-L} \cdots w_{i-1} \in \mathcal{V}^L$,

$$\text{Softmax}(\text{Logits}(\underline{w})) = \exp(l_i) / \sum_{j=1}^{N} \exp(l_j)$$

transforms the output $\text{Logits}(\underline{w}) = (l_1, \dots, l_N)$ into a probability measure $p(\cdot|w_{i-L} \cdots w_{i-1})$ over the set of predicted next tokens from $\mathcal{V}$. We refer to the process of sampling directly from the distributions p as *pure-sampling.* For concision we denote $w_{<i} := w_{i-L} \cdots w_{i-1}$.

In practice, during inference, different decoding strategies are used to give finer control over the qualitative nature of the output, for example by affecting the trade off between quality and diversity (Caccia et al., 2019; Zhang et al., 2021). For example, *top-k* sampling (Fan et al., 2018) samples only over a renormalized truncated distribution of the top k tokens with the highest conditional probability. More precisely, given a context $w_{<i}$ and a value of $k > 1$, the associated top-k set $\mathcal{V}_k(w_{<i})$ is the subset of $\mathcal{V}$ consisting of the k tokens assigned highest probability by the measure $p(\cdot|w_{<i})$. Top-k sampling assigns mass zero to all tokens outside of the top-k set, while giving tokens inside the top-k set mass

$$q_k(v_i|w_{<i}) := \frac{p(v_i|w_{<i})}{\sum_{v \in \mathcal{V}_k(w_{<i})} p(v|w_{<i})}. \tag{1}$$

Top-p sampling (Holtzman et al., 2020) may be defined analogously, and we omit for brevity.

Our particular focus is on text generated through temperature sampling (Guo et al., 2017). Given a temperature parameter $\tau \in (0, 1]$, the probability distribution Q_τ resulting from sampling from language model P at temperature τ is given by

$$q_\tau(v_i|w_{<i}) := \frac{(p(v_i|w_{<i}))^{\frac{1}{\tau}}}{\sum_{v \in \mathcal{V}} (p(v|w_{<i}))^{\frac{1}{\tau}}}. \tag{2}$$

Temperature, top-k and nucleus sampling are widely used as methods of reducing the chance of sampling from the tail of the probability distribution $p(\cdot|w_{<i})$ (Meister et al., 2023). While helpful, these decoding strategies may result in statistically significant traces due to distortions in the renormalized conditional distributions that detection algorithms may exploit.

Quantifying Statistical Properties Of Text Recall that, given a sequence of tokens $w_1 \cdots w_T$ that we wish to classify, most existing zero-shot detectors are based on log-likelihood, log-rank, or entropy. We give mathematical definitions of these quantities here to highlight how they compare and contrast to our new quantity, TempNorm, introduced in Section 4.

The *per-token log-likelihood* of the sequence $w_1 \cdots w_T$ is given by

$$\frac{1}{T} \log P(w_1 \cdots w_T) = \frac{1}{T} \log \left(\prod_{i=1}^{T} p(w_i|w_{<i}) \right).$$

Note that the often used metric *perplexity* is the exponential of the negative per-token log-likelihood. The *rank* $r(v_i|w_{<i})$ of the token v_i given context $w_{<i}$ is the position of token v_i in the list of tokens ordered by decreasing probability $p(v|w_{<i})$, with the most likely token having rank 1. Analogously, the *per-token log-rank* of $w_1 \cdots w_T$ is given by

$$\frac{1}{T} \log r(w_1 \cdots w_T) = \frac{1}{T} \log \left(\prod_{i=1}^{T} r(w_i|w_{<i}) \right).$$

Finally, the *per-token entropy* is given by

$$\frac{1}{T}H(w_1 \cdots w_T) = \frac{1}{T}\sum_{i=1}^{T}\left(\sum_{v\in\mathcal{V}} p(v|w_{<i})\log p(v|_{w_{<i}})\right).$$

3 RELATED WORK

There are two main approaches to the detection of machine-generated text. The first is to train a supervised classifier, which are often excellent in domain but their performance can degrade heavily when the language model or text domain changes (Bao et al., 2024). Some recent works have begun to address this challenge (Li et al., 2024; Zhang et al., 2024), though still face the limitation of requiring training processes and data. The second, which we study in this article, is the zero-shot approach. State-of-the-art zero-shot approaches are discussed below. We also mention watermarking, which is the practice of modifying the output of a language model in order to leave specific statistical traces which are easy to detect, see, for example, Kirchenbauer et al. (2023). Watermarking works well, but only when incorporated directly into the language model. For a survey of efforts to detect machine-generated text see Ghosal et al. (2023).

DetectGPT (Mitchell et al., 2023) evaluates a text $w_1 \cdots w_T$ by first using a mask-filling model to generate perturbations $\tilde{w_1} \cdots \tilde{w_T}$ of the original text which still convey the same meaning but are differently expressed. They compute the mean log-likelihood $\overline{\mu}$ of these texts, along with the standard deviation σ, and assign the text a score $(P(w_1 \cdots w_T) - \overline{\mu})/\sigma$. Thus, rather than comparing the log-likelihood of $w_1 \cdots w_T$ to a global threshold, they compare it to the log-likelihood of similar texts. DetectGPT marked a very significant upgrade to vanilla log-likelihood in the detection of machine-generated text.

NPR (Su et al., 2023) translates the main idea of DetectGPT to log-rank. The method computes the ratio of the log-rank of $w_1 \cdots w_T$ to the log rank of perturbations of the text, using the same perturbation model as DetectGPT.

Fast-DetectGPT (analytic version, see Bao et al. (2024) equation (5)) works the same way as DetectGPT except that, instead of using a mask-filling model, alternative samples are generated with the surrogate language model used for testing. In particular, $\tilde{\mu} = \frac{1}{T}\sum_{i=1}^{T}\sum_{v\in\mathcal{V}} p(v|w_{<i})\log(p(v|w_{<i}))$ here. As far as we understand, Fast-DetectGPT is currently the best-performing method for detecting machine-generated text.

FourierGPT (Xu et al., 2024) and FACE (Yang et al., 2023) are two promising techniques based on computing discrete Fourier transforms over sequences of log-likelihoods. These methods have the capability to extract subtle signals linked to periodicity, but also face a challenge in how to incorporate the computed spectra in a zero-shot classification test. FACE uses statistics such as Spectral Overlap or Spearman's correlation to compare spectra of collections of human and machine text at the distributional level, while FourierGPT forms a heuristic zero-shot test by summing the first k coefficients, where k is a parameter tuned over a validation set. Any such tuning can be considered training and may place these methods slightly outside the zero-shot regime.

Finally, Persistent Homology Dimension (PHD) (Tulchinskii et al., 2024) evaluates a text by embedding it and computing the persistent homology dimension of the resulting embedded set. It seems to be very effective over long texts, yet the reasons for its effectiveness remain poorly understood. This technique is a notable exception in that it is not based on log-likelihood, log-rank or entropy.

4 TEMPTEST: ZERO-SHOT MACHINE-GENERATED TEXT DETECTION

The denominators of equations (2) and (1), which are not independent of $w_1 \cdots w_T$, represent unusual ways of normalizing probability. They normalize conditional next-token probabilities rather than normalizing the joint probability of a string. This is computationally very efficient but theoretically suboptimal, as discussed later. We term this effect *local normalization distortion* and study it in future work. One should expect this method of normalization to leave statistical traces that we can detect, and we empirically demonstrate this in Section 5. In particular, these observations allow us to use the denominator of equation (2), which we term TempNorm, as an additional metric to help assess whether a text has been generated using temperature sampling.

Definition 4.1 *The temperature-τ per-token* log-*TempNorm of a sequence $\underline{w} = w_1 \cdots w_T$ is given by*

$$\frac{1}{T}\log \epsilon_\tau(\underline{w}) = \frac{1}{T}\sum_{i=1}^{T}\log\left(\sum_{v\in\mathcal{V}}(p(v|w_{<i})^{\frac{1}{\tau}})\right)$$

In Figure 1 we plot the pair

$$\left(\frac{1}{T}\log P(\underline{w}), \frac{1}{T}\log \epsilon_\tau(\underline{w})\right)$$

and, by considering the orthogonal projection onto the x-axis, we see per token log-likelihood fails to effectively distinguish between the human and machine-generated texts. In contrast, adding in per-token log-TempNorm as the second coordinate makes it easy to draw a linear decision boundary dividing human texts and those generated by machine at temperature τ. In TempTest we set the dividing line to have slope $\frac{1}{\tau} - 1$ to construct a one-dimensional test. This value is theoretically justified in Section 4.1.

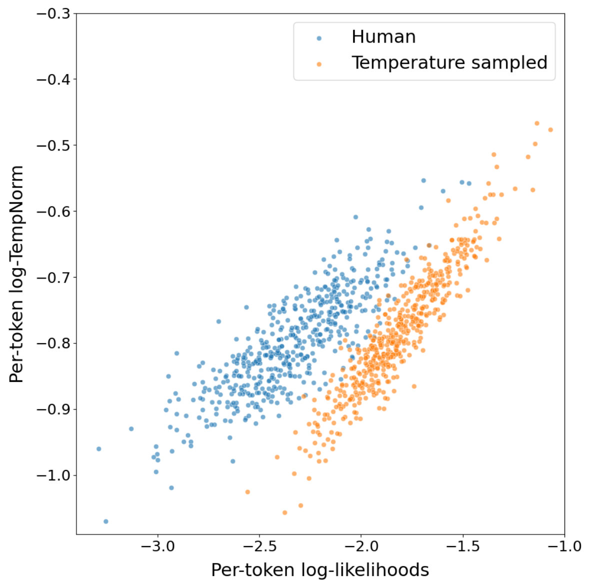

Figure 1: Relationship Between Per-token Log-TempNorm and Per-token Log-likelihood. Observe that per-token log-likelihood alone does a poor job of distinguishing between these sets of 300-token human-written and Llama 3.1 temperature sampled text ($\tau = 0.8$), as indicated by overlap in the orthogonal projection onto the x-axis. Adding a second coordinate displaying per-token log-TempNorm allows for linear separation of the texts.

Definition 4.2 *Given a sequence $\underline{w} = w_1 \cdots w_T$, we define the TempTest score of $\underline{w}$ by*

$$\text{TempTest}(\underline{w}) := \frac{1}{T}\left(\log \epsilon_\tau(\underline{w}) - \left(\frac{1}{\tau} - 1\right)\log P(\underline{w})\right).$$

In Section 4.1 we justify theoretically that, for sequences $\underline{w}_{\text{pure}}$ and $\underline{w}_\tau$ generated by pure-sampling and temperature τ sampling from P respectively, one should expect that $\text{TempTest}(\underline{w}_{\text{pure}}) > 0$ and $\text{TempTest}(\underline{w}_\tau) < 0$. This is a strength compared to methods such as Fast-DetectGPT, where the statistic is not centered on a threshold of 0.

We further show experimentally that the TempTest scores of human-written text are closer to those of pure-sampled text than temperature-sampled text and that TempTest is effective at distinguishing between human-written and temperature-sampled text. We stress that 0 may not be the optimal threshold for distinguishing between human-written text and temperature-sampled text, particularly in the black box setting where a different language model is used for scoring, but empirically, we have found it to be a good estimate; for example, see Section 5.6.

4.1 Theoretical Justification

We give two explanations as to why TempTest is effective at distinguishing between pure-sampled machine-generated text and temperature-sampled machine-generated text. First, a rigorous derivation from a Bayesian perspective, and second, an intuitive justification based on ergodic theory. These also explain our choice of parameter $\frac{1}{\tau} - 1$ in Definition 4.2.

We do not have a theoretical explanation as to why TempTest scores for human-written text are closer to those of pure-sampled text than those of temperature-sampled text. Instead, we rely on the intuition that the manner in which humans generate text does not resemble temperature or top-k sampling, and so human-written text should not suffer from local normalization distortion. This hypothesis is supported by our experimental results, which show very clearly that TempTest is effective.

A Bayesian Perspective Suppose we have a language model P and flip a fair coin. If the outcome is heads, then we generate a text according to P (pure-sampling); if it is tails, we generate a text $\underline{w}$ according to Q_τ (temperature sampling). Based only on knowledge of $\underline{w}$, the detection problem is equivalent to deciding whether we think the coin came down heads or tails.

Let A be the event that the coin came down tails and the text was generated by Q_τ. Let B be the event that the generated text is $\underline{w}$. Bayes' rule tells us that

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A)}{\mathbb{P}(B)}\mathbb{P}(B|A).$$

We want to compute $\mathbb{P}(A|B)$, the probability that the generation method used temperature sampling, given the text. We know $\mathbb{P}(A) = \frac{1}{2}$.

$\mathbb{P}(B)$, the probability that our generation method produced text $\underline{w}$, satisfies

$$\begin{aligned}\mathbb{P}(B) &= \frac{1}{2}P(\underline{w}) + \frac{1}{2}Q(\underline{w}) \\ &= \frac{1}{2}P(\underline{w}) + \frac{1}{2}\frac{P(\underline{w})^{\frac{1}{\tau}}}{\epsilon_\tau(\underline{w})}.\end{aligned}$$

Finally, $\mathbb{P}(B|A) = Q(\underline{w}) = P(\underline{w})^{\frac{1}{\tau}}/\epsilon_\tau(\underline{w})$.

So $\mathbb{P}(A|B)$, the probability that $\underline{w}$ was generated by temperature sampling, is given by

$$\begin{aligned}\mathbb{P}(A|B) &= \frac{1/2}{\frac{1}{2}\left(P(\underline{w}) + \frac{P(\underline{w})^{\frac{1}{\tau}}}{\epsilon_\tau(\underline{w})}\right)}\frac{P(\underline{w})^{\frac{1}{\tau}}}{\epsilon_\tau(\underline{w})} \\ &= \frac{1}{P(\underline{w})^{1-\frac{1}{\tau}}\epsilon_\tau(\underline{w}) + 1}.\end{aligned}$$

In particular, for a threshold $C \in (0,1)$, we conclude that $\mathbb{P}(A|B) > C$ if and only if

$$\epsilon_\tau(\underline{w}) < \left(\frac{1}{C} - 1\right)\frac{1}{P(\underline{w})^{1-\frac{1}{\tau}}}.$$

Using threshold $C = 1/2$, taking logs, and dividing by T, gives that $\mathbb{P}(A|B) > \frac{1}{2}$ if and only if

$$\frac{1}{T}\log \epsilon_\tau(\underline{w}) < \left(\frac{1}{\tau} - 1\right)\frac{1}{T}\log P(\underline{w}).$$

This yields our formulation of TempTest.

Intuition From Ergodic Theory Given a value λ, what do texts generated by pure-sampling of per-token log-likelihood λ look like? The mathematics is slightly cleaner if we consider the limit as passage length T tends to infinity, and so consider the set

$$\mathcal{V}_\lambda := \{\underline{w} \in \mathcal{V}^{\mathbb{N}} : \lim_{T\to\infty}\frac{1}{T}\log p(w_1\cdots w_T) = \lambda\}.$$

Since passages of equal log-likelihood are equally likely under pure-sampling, we are really asking what the measure of maximal Kolmogorov-Sinai entropy on $\mathcal{V}_\lambda$ is. A fact well known to ergodic theorists and fractal geometers is that there exists a temperature τ such that this measure of maximal entropy is given by the Gibbs measure on $\mathcal{V}^{\mathbb{N}}$ at temperature τ, see, for example, the section on the multifractal analysis of ergodic averages in Falconer (2014). This measure picks each string $w_1\cdots w_T$ with probability proportional to $p(w_1\cdots w_T)^{\frac{1}{\tau}}$. That is, these strings are distributed as those of temperature sampling if we were to remove the local normalization distortion.

If we are then to ask whether a string of log-likelihood λ looks more like pure-sampled strings of log-likelihood λ or temperature τ sampled strings of log-likelihood λ, we are effectively asking about the local normalization distortion (TempNorm) of the string. This motivates the study of TempNorm and the development of TempTest.

5 EXPERIMENTS

According to TempTest, does human-written text look more like pure-sampled text or temperature-sampled text of the same log-likelihood?

To answer this question, we consider three core settings: white box, gray box, and black box. In the white box setting we have full access to the language model and temperature used to generate text and use this language model in the evaluation. In the gray box setting we have full access to the language model but do not know the temperature used in generation. In the black box setting we use a different language model for the evaluation of the text. For real-world applications it is important to assess fairness and robustness of new methods. Therefore, we test the degree of bias towards non-native speakers in TempTest, following a setup of Tulchinskii et al. (2024), and robustness to paraphrasing attacks using Dipper (Krishna et al., 2024).

5.1 Setup

We follow Mitchell et al. (2023) by evaluating across three text datasets: news articles from XSum (Narayan et al., 2018), Wikipedia articles from SQuAD (Rajpurkar et al., 2016), and stories from the Reddit WritingPrompts (Writing) dataset (Fan et al., 2018). In the black box setting, we also use generations by GPT3.5-Turbo and GPT-4 (Achiam et al., 2023) supplied by Fast-DetectGPT (Bao et al., 2024) and include the PubMedQA dataset (Jin et al., 2019). Given a large language model, for each sample of human text we generated a synthetic version using the first 30 tokens as context, except for PubMedQA where we take the whole question as context. To ensure a fair comparison, for both human and machine texts we then discarded the shared context to only consider the real and synthetic completions. Hereafter, sample size refers to the size of the completion, not including the original context.

For white box detection, we evaluate using language models Llama 3.1-8B (Dubey et al., 2024), GPT2-XL (Radford et al., 2019), GPT-Neo-2.7B (Black et al., 2021), GPT-J-6B (Wang and Komatsuzaki, 2021) and OPT-2.7B (Zhang et al., 2022). The choice of detection models largely follows Bao et al. (2024).

All methods typically improve as the number of tokens in the sample to be tested increases. Indeed, state-of-the-art methods are nearing perfect AUROC scores at 200 to 300 tokens. Therefore, we primarily consider the most challenging scenario of small text snippets, typically using inputs of length 50 tokens. This case highlights the strength of the signal and statistical power of the respective methods. For completeness, we include a comparison over longer texts too in Section 5.5.

As is standard, for example see Mitchell et al. (2023); Bao et al. (2024), we use the AUROC measure to evaluate the performance of detectors. However, we note one practical strength of our method, not captured by this choice of metric, is a natural and theoretically justified decision threshold. Alongside recent methods detailed in Section 3, we also use entropy, rank, and log-rank as baselines.

For all baseline methods, we have used the author's own implementations, drawing on the repository supplied with Bao et al. (2024). Supporting code is available on GitHub[1]. Experiments were run on a cluster of 8 A100 GPUs.

5.2 White Box

Table 1 details comparisons of TempTest against 6 baseline methods across 5 large language models and 3 datasets. TempTest was typically comparable or superior to the next best baseline with an average rank of 1.2, compared to the Fast-DetectGPT which had an average rank of 1.6.

5.3 Gray Box

Figure 2 demonstrates that TempTest is very robust to choosing the wrong scoring temperature. That is, if the generation temperature is not known, as in a realistic scenario, choosing arbitrarily should suffice. For example, if auditing a selection of creative writing essays, an educated guess might suppose that counterfeits are using a relatively high temperature such as 0.7 to 0.8 for greater generated diversity.

5.4 Black Box

Table 2 details comparisons of TempTest against 6 baseline methods across 4 large language models and 3 datasets. TempTest was typically comparable or superior to the next best baseline with an average rank of 1.3, compared to Fast-DetectGPT, which had an average rank of 1.6. In addition, in Figure 3 we show TempTest is stronger than or equal to Fast-DetectGPT over 3 sample sizes on synthetic data from GPT-3.5-Turbo and GPT-4. We also benchmarked PHD (Tulchinskii et al., 2024), and found it to be comparable with middle-ranked baselines, see Appendix 9.3.

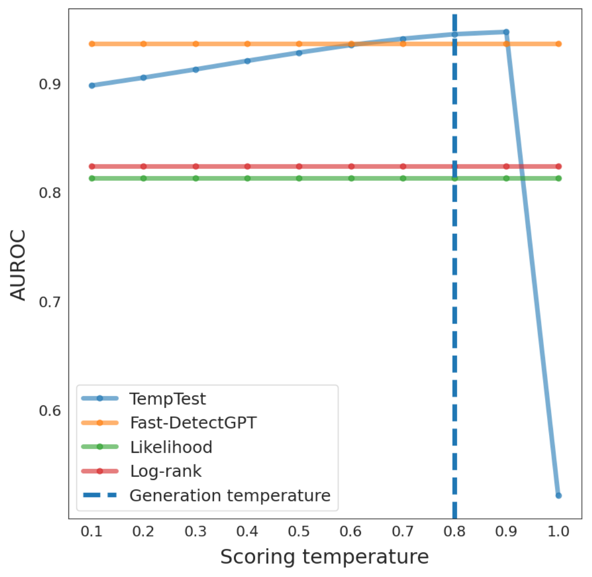

Figure 2: Performance Across Scoring Temperatures. TempTest is robust to the particular choice of temperature used for scoring, even if it is significantly different from the ground-truth temperature used for generation. As expected, for $\tau = 1$ the AUROC drops to 0.5, as this case equivalent to pure-sampling. Meta Llama 3.1-8B is used for generation and scoring, and the ground-truth generation temperature is 0.8.

5.5 Effect Of Sample Size

Our white and black box experiments focus on a sample size of 50, a challenging task that highlights statistical power and helps to separate methods. As sample size increases, we typically see all methods perform better, and ultimately converge. Figure 4 shows AUROCs averaged across all datasets and models used in Table 1. TempTest outperforms all baselines at low sample sizes and is marginally better or comparable at higher sample sizes.

5.6 Non-native Speaker Bias

Machine-generated text detection algorithms have been shown to be potentially biased against non-native speakers by classifying them as machines (Liang et al., 2023). To evaluate TempTest in this setting, we followed the broad setup of Liang et al. (2023). This involved comparing detection algorithms on TOEFL

[1]`https://github.com/TMKempton/TempTest`

Table 1: White Box Experiment Results. In this setting the potential generation model and temperature are known at test time. Sample texts of length 50 tokens were used, and $\tau = 0.8$ for generation and scoring.

Model	Llama 3.1			GPT2-XL			GPT-Neo 2.7b			GPT-J 6b			OPT-2.7b			Mean
Dataset	Writing	SQuAD	XSum	Writing	SQuAD	XSum	Writing	SQuAD	XSum	Writing	SQuAD	XSum	Writing	SQuAD	XSum	
Entropy	.516	.606	.573	.520	.616	.612	.508	.634	.624	.506	.645	.620	.516	.600	.569	.578
Log-rank	.815	.697	.683	.887	.862	.846	.883	.822	.838	.882	.780	.821	.887	.814	.842	.824
Likelihood	.806	.655	.662	.888	.839	.851	.885	.794	.844	.883	.747	.825	.879	.795	.842	.813
DetectGPT	.585	.498	.555	.798	.772	.781	.757	.717	.766	.708	.667	.702	.788	.676	.707	.698
DetectLLM	.607	.538	.593	.812	.538	.811	.774	.739	.787	.735	.721	.733	.816	.703	.731	.709
Fast-DetectGPT	.914	.869	.790	.969	**.965**	.966	**.963**	**.954**	**.963**	.959	**.940**	**.949**	.959	.941	.945	.936
TempTest	**.935**	**.893**	**.848**	**.972**	.964	**.967**	**.963**	.951	**.963**	**.960**	.939	**.949**	**.964**	**.946**	**.964**	**.945**

Table 2: Black Box Experiment Results. In this setting the temperature is known, but the generation model, Meta Llama 3.1-8B, is not known at test time. Hence, a variety of alternative scoring models were evaluated. Sample texts of length 50 tokens were used. If the temperature is also unknown and must be estimated, only a small degradation is expected; see Figure 2.

Scoring Model	GPT2-XL			GPT-Neo 2.7b			GPT-J 6b			OPT-2.7b			Mean
Dataset	Writing	SQuAD	XSum	Writing	SQuAD	XSum	Writing	SQuAD	XSum	Writing	SQuAD	XSum	
Entropy	.492	.538	.567	.478	.527	.524	.492	.560	.549	.478	.533	.526	.522
Log-rank	.802	.722	.622	.807	.733	.702	.807	.715	.692	.807	.727	.694	.736
Likelihood	.770	.686	.562	.794	.707	.684	.793	.681	.670	.800	.711	.690	.712
DetectGPT	.575	.566	.513	.587	.553	.567	.572	.508	.548	.566	.561	.583	.558
DetectLLM	.628	.584	.563	.619	.561	.584	.600	.544	.586	.579	.556	.592	.583
Fast-DetectGPT	**.847**	**.797**	.643	**.880**	.825	.754	**.882**	**.827**	.755	.887	.837	.775	.809
TempTest	**.847**	.794	**.647**	.877	**.826**	**.760**	.881	.822	**.758**	**.898**	**.843**	**.831**	**.815**

essays written by non-native speakers with the same essays re-written by GPT-4. GPT-4 was prompted to re-write the essays to sound more like a non-native speaker, see Appendix 8 for details. To obtain concrete predictions from scores, Liang et al. (2023) use decision thresholds determined by the *equal error rate* (EER) on a held-out dataset (Writing (Fan et al., 2018)). For both tuning and scoring we use GPT-Neo 2.7b, the favored model of Fast-DetectGPT. Figure 8 (Appendix 9) shows TempTest has improved overall error rates to the baseline Fast-DetectGPT, and TempTest was less likely to classify the human text as machine-generated. This preliminary result suggests TempTest may be effective in reducing some sources of bias in machine-generated text detection. Although we roughly followed the experimental setup of Tulchinskii et al. (2024) for continuity with the existing literature, it should be noted that this is a small-scale experiment due to the limited available data. We encourage the development of larger datasets, new methods to mitigate bias, and more extensive analysis in a range of experimental settings.

5.7 Robustness To Paraphrasing Attacks

Paraphrasing attacks, using models such as Dipper (Krishna et al., 2024), have been shown to significantly degrade the performance of machine-generated text detectors. Figure 5 shows that TempTest is impacted by such attacks, but is far less susceptible than the best baseline, Fast-DetectGPT. A wider suite of adversarial attacks is investigated in (Liu et al., 2024), and we leave it to future work to verify the effectiveness of TempTest in this context.

6 DISCUSSION

Section 5 empirically validates the effectiveness of TempTest as a zero-shot method for detecting machine-generated text. However, it is reasonable to ask how TempTest and other methods of detecting machine-generated text will fare in the future as language models become more sophisticated. One may expect that methods based on log-likelihood or log-rank will see their performance degrade as the probability distributions that language models output start to resemble human texts more closely, a hypothesis foreshadowed by our results on Llama 3.1. Similarly, we would expect that, when plotting log-likelihood against TempNorm for pure-sampled machine-generated texts, they will start to resemble the plots for human texts more closely. However, the statistical differences between pure-sampled text and temperature-sampled text are not a function of the quality of the language model, and so one should not expect TempTest to become less good at differentiating between human written and temperature-sampled text. Indeed, the main threat to the longevity of our method is that temperature sampling and top-k sampling may go out of fashion.

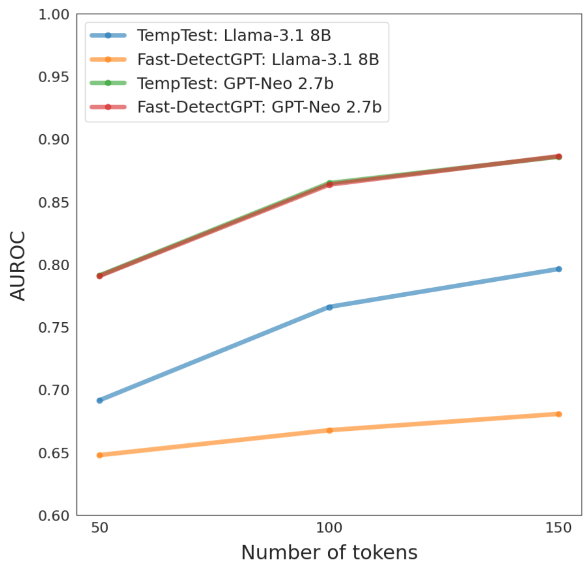

Figure 3: Performance On GPT-3.5 Turbo And GPT-4 Generated Data. TempTest obtains comparable or superior performance to Fast-DetectGPT across a range of passage lengths in an illustrative black-box setting when using GPT-3.5 Turbo or GPT-4 as the generation model. Two scoring models were considered here: Meta Llama 3.1-8B and GPT-Neo 2.7b. A temperature of 0.8 was used for generation and scoring. Results are averaged over Writing, XSum, and PubMedQA completions.

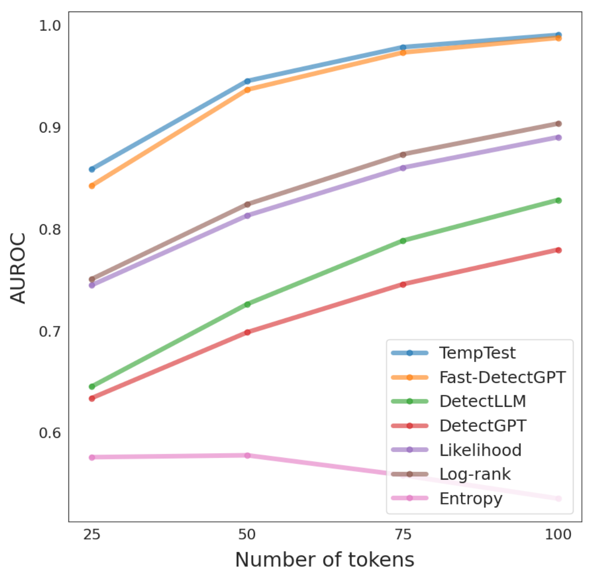

Figure 4: Performance For Varying Context Lengths. AUROCs averaged across all 3 datasets and 5 models used in Table 1 at input text sizes ranging from 25 to 100 tokens.

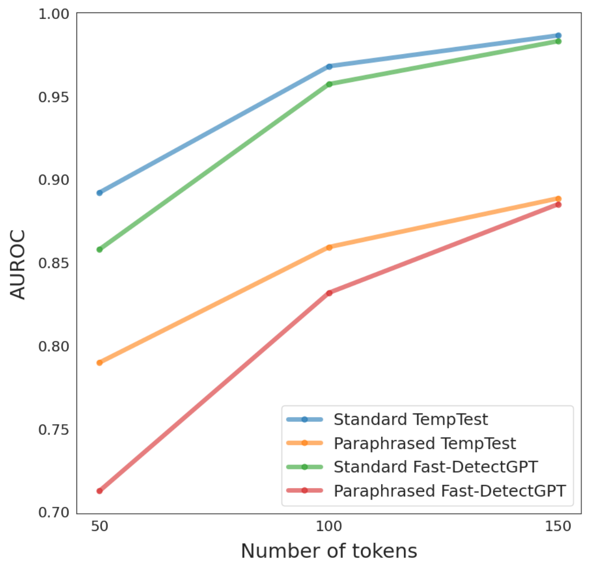

Figure 5: Robustness Of TempTest Under Paraphrasing. TempTest is less susceptible to paraphrasing attacks than the previous art, Fast-DetectGPT, over a range of input text sizes. Reported AUROC values are the average over datasets Writing, XSum, and SQuAD.

7 CONCLUSIONS

Across a wide range of scenarios, we have demonstrated that TempTest is an effective new approach for detecting machine-generated text that is comparable or superior to the current state-of-the-art, with this effect most pronounced for short passage lengths and on modern language models. This test is based on a novel statistical quantity, TempNorm, which quantifies the local normalization distortion of conditional probability measures. Importantly, this provides an extra tool for researchers and practitioners to use when building future detection algorithms.

However, there are drawbacks to our method that present opportunities for future work. First, we consider only English and these empirical investigations should be expanded to include other languages. Second, the distortion we detect is only present in temperature-sampled text and thus is not applicable to text generated through different generation strategies such as top-p or top-k. That said, an analogous method may be derived for top-k, see Appendix 9.2. Moreover, we hope that future work will look at ways of ensembling or unifying different zero-shot methods into a single approach, creating methods that are broadly applicable and exploiting the advantages of each technique. Ultimately, the more diverse our statistical toolbox is for analyzing text, the better we will be able to detect the traces machines leave behind.

Acknowledgements

Computational resources were provided by Featurespace. We also thank our reviewers for their helpful comments.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., and others. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774.*

Ahmed, A. A. A., Aljabouh, A., Donepudi, P. K., and Choi, M. S. (2023). Detecting Fake News Using Machine Learning : A Systematic Literature Review. *arXiv preprint arXiv:2102.04458.*

Bao, G., Zhao, Y., Teng, Z., Yang, L., and Zhang, Y. (2024). Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature. In *The Twelfth International Conference on Learning Representations.*

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

Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., and Charlin, L. (2019). Language GANs Falling Short. In *International Conference on Learning Representations.*

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783.*

Elkhatat, A. M. (2023). Evaluating the authenticity of chatgpt responses: a study on text-matching capabilities. *International Journal for Educational Integrity.*

Falconer, K. (2014). *Fractal Geometry: Mathematical Foundations and Applications.* John Wiley & Sons.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Gehrmann, S., Strobelt, H., and Rush, A. M. (2019). GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Ghosal, S. S., Chakraborty, S., Geiping, J., Huang, F., Manocha, D., and Bedi, A. (2023). A Survey on the Possibilities & Impossibilities of AI-generated Text Detection. *Transactions on Machine Learning Research.*

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations.*

Jin, Q., Dhingra, B., Liu, Z., Cohen, W., and Lu, X. (2019). 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)*, pages 2567–2577.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. (2023). A Watermark for Large Language Models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. (2024). Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

Li, Y., Wang, Z., Cui, L., Bi, W., Shi, S., and Zhang, Y. (2024). Spotting AI‘s Touch: Identifying LLM-Paraphrased Spans in Text. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7088–7107, Bangkok, Thailand. Association for Computational Linguistics.

Liang, W., Yuksekgonul, M., Mao, Y., Wu, E., and Zou, J. Y. (2023). GPT detectors are biased against non-native English writers. *Patterns*, 4.

Liu, Z., Cong, T., He, X., and Li, Q. (2024). On Evaluating The Performance of Watermarked Machine-Generated Texts Under Adversarial Attacks. *arXiv preprint arXiv:2407.04794.*

Meister, C., Pimentel, T., Malagutti, L., Wilcox, E., and Cotterell, R. (2023). On the Efficacy of Sampling Adapters. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1437–1455.

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. (2023). DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Narayan, S., Cohen, S. B., and Lapata, M. (2018). 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*. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Su, J., Zhuo, T., Wang, D., and Nakov, P. (2023). DetectLLM: Leveraging Log Rank Information for Zero-Shot Detection of Machine-Generated Text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

Tulchinskii, E., Kuznetsov, K., Kushnareva, L., Cherniavskii, D., Nikolenko, S., Burnaev, E., Barannikov, S., and Piontkovskaya, I. (2024). Intrinsic Dimension Estimation for Robust Detection of AI-Generated Texts. *Advances in Neural Information Processing Systems*, 36.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in neural information processing systems*, 30.

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

Xu, Y., Wang, Y., An, H., Liu, Z., and Li, Y. (2024). Detecting Subtle Differences between Human and Model Languages Using Spectrum of Relative Likelihood. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10108–10121, Miami, Florida, USA. Association for Computational Linguistics.

Yang, X., Cheng, W., Wu, Y., Petzold, L. R., Wang, W. Y., and Chen, H. (2024). DNA-GPT: Divergent N-Gram Analysis for Training-Free Detection of GPT-Generated Text. In *The Twelfth International Conference on Learning Representations.*

Yang, Z., Yuan, Y., Xu, Y., Zhan, S., Bai, H., and Chen, K. (2023). FACE: Evaluating Natural Language Generation with Fourier Analysis of Cross-Entropy. *Advances in Neural Information Processing Systems*, 36:17038–17056.

Zhang, H., Duckworth, D., Ippolito, D., and Neelakantan, A. (2021). Trading Off Diversity and Quality in Natural Language Generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068.*

Zhang, S., Song, Y., Yang, J., Li, Y., Han, B., and Tan, M. (2024). Detecting Machine-Generated Texts by Multi-Population Aware Optimization for Maximum Mean Discrepancy. In *The Twelfth International Conference on Learning Representations.*

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, see Subsection 9.6.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, see Section 5.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator if your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

SUPPLEMENTARY MATERIALS

8 EXPERIMENTAL DETAILS

We gather here experimental details described elsewhere in the paper.

Usually, when generating machine text, 30 tokens of human text are given as a prompt to the language model, which generates a completion. The exception is the PubMedQA dataset (Jin et al., 2019), which contains questions and answers. In that case, the whole question was given as a prompt to the language model. Machine-generated texts were of length 50 tokens except where stated otherwise, for example Figure 4.

White Box In these experiments, machine-generated texts were generated using the XSum, SQuAD and Writing datasets. The machine-generated texts were generated by Llama 3.1-8B, GPT2-XL, GPT-Neo-2.7B, GPT-J-6B and OPT-2.7B, each at temperature 0.8. In each case, the same language model was used for evaluation, and TempTest was run with the correct temperature of 0.8.

Black Box We ran two separate experiments. In the first, texts were generated using GPT2-XL, GPT-Neo-2.7B, GPT-J-6B and OPT-2.7B at temperature 0.8, with prompts from XSum, SQuAD and Writing. The texts were evaluated using Llama 3.1-8B. For the second set of experiments (Tables 3 and 4), texts were generated using GPT3.5 Turbo and GPT4 at temperature 0.8, with prompts from Writing, PubMedQA and XSum. Llama 3.1-8B and GPT-Neo-2.7B were used as evaluators; see Subsection 8.1 for the rationale behind these scoring models.

Gray box We experimented with the case where the generating model is known but the generating temperature is not. In this case, we generated texts using Llama 3.1-8B at temperature 0.8, and ran TempTest with scoring model Llama 3.1-8B at a variety of temperatures.

8.1 A Comment On The Choice of Scoring Model

The datasets (XSum, PubMedQA, Writing and SQuAD) upon which we evaluate detector performance have been used in many previous works, including state-of-the-art detector FastDetect-GPT. These works evaluated their detectors using a range of different language models as scorers. In order to avoid selecting the scoring language model based on performance on the test set, we took the decision to use Llama 3.1-8B as the primary black-box scoring model, since this is a modern language model which had not previously been used with existing art. The one exception is in our experiment on texts generated by GPT-3.5 Turbo and GPT4. Here, it appears that FastDetect-GPT performs much worse when Llama is used as the detection model, and so we also included results using the best-performing language model for their paper, GPT-Neo 2.7B. See Subsection 9.

8.2 Two Cautionary Tales

We mention two things to be aware of when evaluating the performance of detectors. The first is that, at time of writing, the HuggingFace `GenerationConfig` class has top-k set to 50 by default, and so one must explicitly set top-k to 0 or `None` to deactivate this. At least three papers in the area have reported results for detecting pure-sampled text where the experiment was actually run on top-k ($k = 50$) sampled text as a result of this choice of default. The second thing to be aware of is that several detectors produce results whose expected values are not independent of the length of the text. This is not in itself a problem, provided calibration of the correct classification threshold is done separately for each value of text length T. It does, however, mean that AUROC values computed over datasets where the human and machine-generated text have different lengths can be highly misleading. Some papers in the area study machine texts of 200 or 300 tokens, whereas the human written texts vary naturally in length but are typically much lower, which can heavily skew the results.

Table 3: Detecting GPT-3.5 And GPT-4 Generated Text With Scoring Model GPT-Neo 2.7b. In this experiment the generation temperature was 0.8 and the text length was 50 tokens. TempTest was evaluated with scoring temperatures, denoted in parentheses, between 0.6 and 0.9.

Generation Model	GPT-3.5 Turbo			GPT-4			Mean
Dataset	Writing	PubMedQA	XSum	Writing	PubMedQA	XSum	
Fast-DetectGPT	.861	.785	**.895**	.727	.694	**.783**	**.791**
TempTest ($\tau = 0.6$)	.861	**.815**	.878	.704	**.713**	.741	.785
TempTest ($\tau = 0.7$)	.863	.809	.883	.715	.708	.754	.789
TempTest ($\tau = 0.8$)	**.866**	.800	.889	.725	.701	.767	**.791**
TempTest ($\tau = 0.9$)	**.866**	.785	.893	**.733**	.694	.777	**.791**

Table 4: Detecting GPT-3.5 And GPT-4 Generated Text With Scoring Model Meta Llama 3.1-8B. In this experiment the generation temperature was 0.8 and the text length was 50 tokens. TempTest was evaluated with scoring temperatures, denoted in parentheses, between 0.6 and 0.9.

Generation Model	GPT-3.5 Turbo			GPT-4			Mean
Dataset	Writing	PubMedQA	XSum	Writing	PubMedQA	XSum	
Fast-DetectGPT	.705	.738	.627	**.698**	.593	.526	.648
TempTest ($\tau = 0.6$)	**.796**	**.812**	**.732**	.697	**.667**	.570	**.712**
TempTest ($\tau = 0.7$)	.777	.798	.725	.696	.650	.574	.703
TempTest ($\tau = 0.8$)	.755	.779	.715	.693	.631	**.577**	.692
TempTest ($\tau = 0.9$)	.726	.757	.700	.689	.611	**.577**	.677

9 ADDITIONAL RESULTS AND METHODS

9.1 Further experiments on GPT-3.5 and GPT-4

In Tables 3 and 4 we present the full (disaggregated) results used in Figure 3, where we compared the performance of TempTest to FastDetect-GPT for detecting text generated by GPT-3.5 Turbo and GPT-4 at temperature 0.8.

9.2 Detecting Top-k Generated Text

We describe a test analogous to TempTest for detecting text which has been generated by a language model using top-k sampling.

The simplest white box method for detecting whether a text, or a portion of a text, has been generated by a given language model using top-k sampling is to scan through the text, looking for passages where each token lies in the top-k set. In Gehrmann et al. (2019) this was used to produce a color overlay for passages of text, where tokens in the top-10 set were colored green, tokens in the top-100 set were colored yellow and tokens in the top-1000 set were colored red.

As always, one runs the risk of false positives. Scanning through human-written texts from the writing prompts dataset, we find 61 of the 500 essays contain at least one passage of at least 30 consecutive tokens, which lie in the top-50 set. This begs the question of whether one can distinguish between human-written passages which lie in the top-50 set and passages generated by top-50 sampling.

Our method follows the reasoning of TempTest. Recall that the Bayesian justification for TempTest was not able to compare human-written text with temperature-sampled text; instead there was a rigorously justified method for assessing whether a text was more likely generated by pure sampling or temperature sampling, and then experimental evidence that this method was also effective at distinguishing between human-written and temperature-sampled text. Similarly, given a passage of text that lies in the top-k set, we give a rigorous justification of a method to determine whether it was more likely generated by top-k sampling or pure sampling.

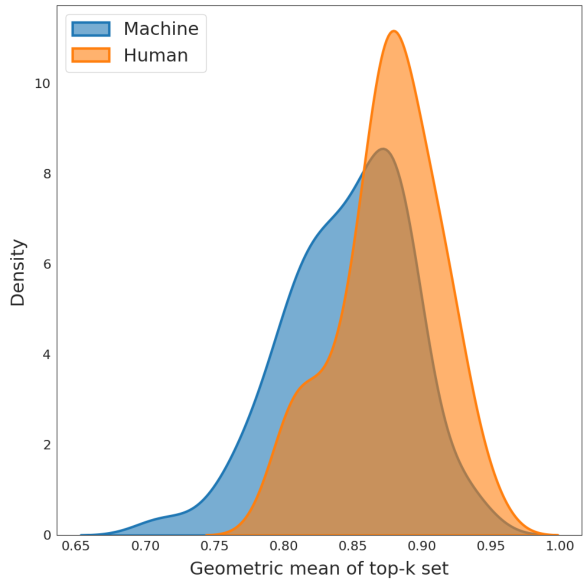

Figure 6: Comparing Geometric Means Of Machine And Human Text. This density plot highlights a discrepancy between the geometric mean of the size of the top-k set for human-written passages that are coincidentally in the top-k set and machine-generated passages that used top-k sampling.

We recall that the denominators in equation (1) are non-constant, and so top-k sampling does not sample strings $w_1 \cdots w_T$ from the top-k set with probability proportional to $p(w_1 \cdots w_T)$. This means that passages $w_i \cdots w_l$ for which, on average, the top-k set is rather small are (relatively) much more likely under top-k sampling than they would have been by sampling directly from p.

Another way of viewing this is that it is much easier to accidentally generate a passage of text for which each token lies in the corresponding top-50 set when these top-50 sets have a large mass. We reproduce below the argument of Section 4.1 for a careful Bayesian justification of this claim. This leads us to the following:

Previous test (in the spirit of Gehrmann et al. (2019)) Scan the text to identify passages of at least thirty consecutive tokens lying in the top-50 set. Declare such passages suspicious

New test Scan the text to identify passages of at least thirty consecutive tokens lying in the top-50 set. Given such a passage $w_j \cdots w_l$, compute the geometric mean of the size of top-50 sets arising from the passage, i.e.

$$\exp\left(\frac{1}{l-j+1}\log\left(\prod_{i=j}^{l}\sum_{v\in\mathcal{V}_k(w_{<i})} p(v|w_{<i})\right)\right).$$

If this mean size is below some threshold C, declare the passage suspicious.

We apply these tests to human-written texts in the Writing dataset. Wherever a human-written text $\underline{w}$ has a passage $w_i \cdots w_j$ of at least thirty tokens in the top-50 set, we use context $w_1 \cdots w_{i-1}$ to prompt Llama 3.1 with top-50 sampling to generate an alternative passage of the same length. We then compare in Figure 6 the geometric mean of the size of the top-50 set across these human and machine passages.

Our new test achieves an AUROC of 0.695 in distinguishing human-written passages, which would have been declared suspicious by the old test from top-50 sampling machine-generated passages. Thus, even on passages of length thirty tokens, the extra information added by top-k-mass significantly increases our ability to reliably detect passages generated by top-k sampling.

9.2.1 The Bayesian Perspective On Our Top-k Test

Let us fix a language model P and a natural number k. We wish to compare the probabilities that a text was generated by pure sampling or top-k sampling, conditioned on the fact that this text is known to lie in the top-k set. Equivalently, we are asking for the relative probabilities that the text was generated by top-k sampling or rejection sampling, where the rejection sampling comes from pure sampling repeatedly until one generates a sample which lies in the top-k set. This rejection sampling assigns mass $P(\underline{w})/C$ to elements of the top-k set, where C is some unknown constant.

Suppose that we flip a fair coin and according to the outcome we generate a text according to rejection-sampling P/C (if the coin came down heads) or top-k sampling Q_k (if the coin came down tails). Suppose we are given a text $\underline{w} = w_1 \cdots w_T$ which lies in the top-k set. Based only on knowledge of $\underline{w}$, the detection problem is equivalent to deciding whether we think the coin came down heads or tails.

Let A be the event that the coin came down tails and the text was generated by top-k sampling. Let B be the event that the generated text is $\underline{w}$. Bayes' rule tells us that

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A)}{\mathbb{P}(B)}\mathbb{P}(B|A).$$

We want to compute $\mathbb{P}(A|B)$, the probability that the generation method used top-k sampling, given the text. We know $\mathbb{P}(A) = \frac{1}{2}$.

$\mathbb{P}(B)$, the probability that our generation method produced text $\underline{w}$, satisfies

$$\begin{aligned}\mathbb{P}(B) &= \frac{1}{2C}P(\underline{w}) + \frac{1}{2}Q_k(\underline{w}) \\ &= \frac{1}{2C}P(\underline{w}) + \frac{1}{2}\frac{P(\underline{w})}{\epsilon_k(\underline{w})}\end{aligned}$$

where

$$\epsilon_k(\underline{w}) := \prod_{i=1}^{T} \sum_{v \in \mathcal{V}_k(w_{<i})} p(v|w_{<i})$$

is the product of the masses of the top-k sets at each time $i \in \{1, \cdots T\}$. Finally, $\mathbb{P}(B|A) = Q_k(\underline{w}) = P(\underline{w})/\epsilon_k(\underline{w})$.

So $\mathbb{P}(A|B)$, the probability that $\underline{w}$ was generated by top-k sampling, is given by

$$\begin{aligned}\mathbb{P}(A|B) &= \frac{1/2}{\frac{1}{2}\left(\frac{P(\underline{w})}{C} + \frac{P(\underline{w})}{\epsilon_k(\underline{w})}\right)}\frac{P(\underline{w})}{\epsilon_k(\underline{w})} \\ &= \frac{1}{1 + \frac{\epsilon_k(\underline{w})}{C}}.\end{aligned}$$

In particular, since C is a fixed constant independent of $\underline{w}$, we see that the conditional probability that the text was generated by top-k sampling depends only on the product of the masses of the top-k sets along the sequence $\underline{w}$, as required.

9.3 PHD Results

As mentioned in Subsection 5.4, we benchmark TempTest against PHD, a method for detecting machine-generated text based on persistent homology dimension Tulchinskii et al. (2024); see Figure 7. Results are always in the black box setting, since PHD uses an embedding model for evaluation rather than a standard language model. Results are aggregated over Writing, XSum, and SQuAD.

9.4 Experiment On TOEFL Data

Figure 8 gives the full confusion matrix for the experiment described in Section 5.6.

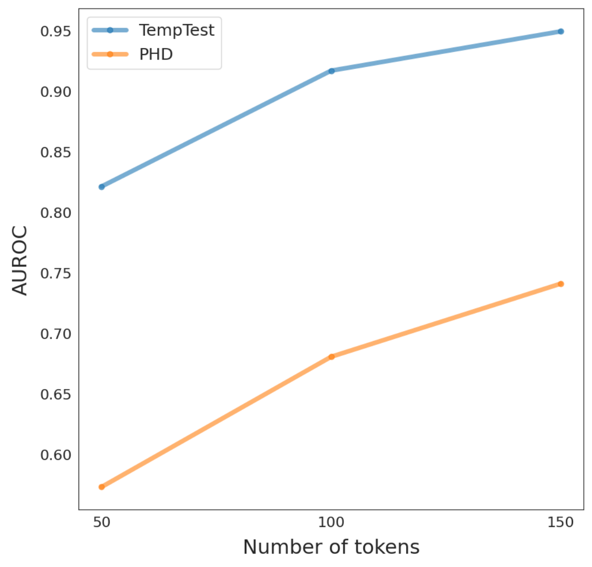

Figure 7: PHD And TempTest In A Black Box Setting. Roberta-base is used for PHD as the embedding model as recommended in Tulchinskii et al. (2024), while GPT-Neo 2.7B is used for scoring with TempTest. Temperature 0.8 was used throughout. Results are aggregated over Writing, XSum, and SQuAD.

9.5 Nucleus Sampling

There is a natural analogue of our top-k approach to detect text generated using nucleus (top-p) sampling (Holtzman et al., 2020). Nucleus sampling is similar to top-k sampling, except that instead of restricting to the k most likely tokens, it restricts to the smallest set of most likely tokens whose cumulative probability exceeds a threshold p. If the cumulative probability was always exactly p, then there would be no local normalization distortion, but since this is not true, one can follow the method of Section 9.2 with the mass of the top-k set replaced with the mass of the top-p set. Unfortunately, the signal one is hoping to detect is much weaker here, and so we were not able to produce analogous empirical results.

9.6 Algorithmic Complexity

The dominant and most costly part of TempTest is performing a single forward pass with the scoring language model to compute the conditional probabilities. Therefore, TempTest inherits the chosen model's time and memory complexity.

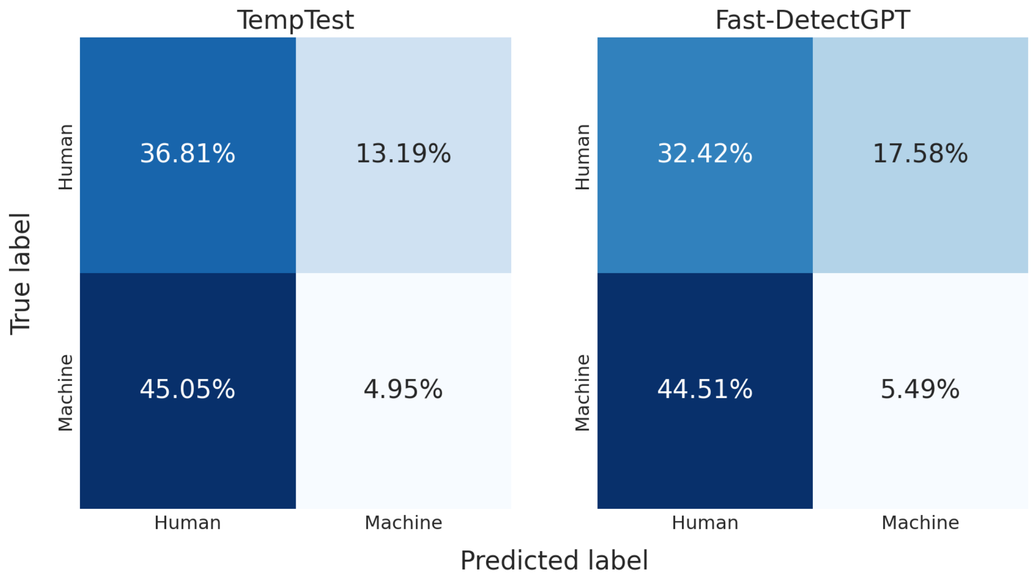

Figure 8: TempTest Reduces Bias Against Non-native Speakers. Confusion matrices for performance of TempTest and Fast-DetectGPT on a dataset of TOEFL essays by non-native speakers and machine-polished adaptations. Classification was done over 50 tokens and at an EER threshold tuned via an independent dataset (Writing). The model used for scoring and tuning the EER was GPT-Neo 2.7b, the empirically favored model of Fast-DetectGPT. TempTest has improved overall performance, and notably, the number of human essays flagged as machine is reduced. This preliminary result suggests TempTest may be effective at reducing some sources of bias in machine-generated text detection. Both methods perform poorly on the machine-polished text, indicating further research should be done to combat this form of paraphrasing attack.