FastDetectGPT: Understanding Sampling-based Zero-shot AI-generated Text Detection

Jose Garcia[1], Krystof Bobek[1], Mara Dragomir[1], Mahdi Rahimi[1], Salvador Torpes[1]

[1]University of Amsterdam

Contact: { jose.garcia.carrillo, krystof.bobek, mara.dragomir, mahdi.rahimi, salvador.baptista.torpes } @student.uva.nl

Abstract

DetectGPT introduced a sampling-based zero shot detector that exploits local curvature of the log likelihood surface to distinguish machine-generated from human written text. Fast DetectGPT later proposed a faster method that maintains strong performance with reduced computational cost. In this project we reproduce Fast DetectGPT as baselines with new source models (R1-8B, Phi-2, Mistral-7B, GPT-4o mini). We evaluate on three datasets: Xsum, which has already been used in the previous work, and HC3, which is a new dataset and Scottish Gaelic as an underrepresented language. We introduce a novel adversarial setting: the Mid-k perturbation attack that forces the generator to avoid high-probability tokens at random positions. Our results show that Fast DetectGPT remains effective across diverse models and datasets, but Mid-K perturbation can trick the detector.

1 Introduction

Large language models (LLMs) such as ChatGPT (Brown et al., 2020), have revolutionized content creation across industries, from journalism to academic research. However, their unprecedented fluency has raised significant concerns about misuse in generating fake news (Ahmed et al., 2021), fraudulent reviews (Adelani et al., 2019), and facilitating plagiarism (Lee et al., 2023). The challenge of distinguishing machine-generated from human-written text has become increasingly difficult, even for experts (Shahid et al., 2022), creating the necessity for reliable automated detection methods.

Current detection approaches fall into two categories: supervised classifiers (LLMs fine-tuned on human and AI-written data to learn how to classify them) (Mitrović et al., 2023) and zero-shot methods (Su et al., 2023) (do not require any fine-tuning, only largely trained models). While supervised methods perform well within their training domains, they suffer from poor generalization across different domains and models. Zero-shot detectors offer better domain robustness by leveraging pre-trained language models without fine-tuning, relying on universal features that generalize across contexts.

Zero-shot detectors are LLM-based algorithms that identify machine-generated text by looking at its inherent properties, e.g., their rank, likelihood, entropy. We assume that machine-generated text has different statistical properties than human-written text, allowing us to distinguish between the two. For instance, the log-likelihood of machine-generated text is often higher than that of human-written text, as LLMs tend to produce more probable sequences when decoding. Hence, by desiging a simple algorithm that compares an internal property of a candidate text with a threshold, we can classify it as machine-generated or human-written.

2 Methodology

We frame machine-generated text detection as a zero-shot binary classification problem. given a candidate passage, determine whether it was written by a human or generated by a language model. Our methodology builds on two recent approaches, DetectGPT and Fast-DetectGPT, and extends them to new models, datasets, and an adversarial attack.

2.1 Datasets

The models are trained on synthetic datasets generated by prompting the source model with the first 30 tokens of sentences from human-written datasets. The source model then generates a continuation up to an average of 230 tokens, producing machine-generated text samples.

Our implementation will use two human-written datasets: XSum (Narayan et al., 2018), a large collection of documents and their respective summaries, and HC3 (Guo et al., 2023), which consists of pairs of human and GPT 3.5 generated answers to corresponding questions (for which only the human answers are used).

2.2 Methods

Baseline Methods focus solely on comparing internal model properties of candidate passages with a threshold to classify them as human or machine-generated.

DetectGPT (Mitchell et al., 2023), introduced a sampling-based criterion for detection. The key hypothesis is that model generated text tends to lie in regions of negative curvature of the log-likelihood surface of the source model. DetectGPT generates multiple perturbations of a candidate passage using a masked language model such as T5. It then compares the log probability of the original passage with those of its perturbations under the source model. If the original passage consistently receives a higher log probability than its perturbations, it is likely machine-generated. However, this approach requires scoring a large number of perturbations per passage (often 100+), making it computationally expensive.

Fast-DetectGPT replaces DetectGPT's perturb-and-rescore pipeline with a single-pass estimate of a conditional probability curvature. Framed as sequential token selection, the hypothesis is that models approximate the modal human distribution, selecting safer, higher-probability tokens, in contrast to individuals whose distinctive styles produce less common selections. Consequently, machine-written passages tend to appear near local maxima of the conditional probability function, yielding positive curvature, whereas human text yields curvature near zero. Operationally, Fast-DetectGPT samples alternative token choices independently at each position conditioned on the observed prefix, computes their conditional probabilities under a scoring model, and aggregates these scores into a curvature statistic. Crucially, all alternatives are produced in one sampling call and scored in one batched forward pass, eliminating the need to evaluate multiple perturbations per passage. The resulting detector achieves strong accuracy while increasing efficiency significantly.

$$\sum_i (\log p_\theta(\tilde{x}_i) - \tilde{\mu})/\tilde{\sigma} > \epsilon \rightarrow x \text{ is AI-generated} \tag{1}$$

In eq. (1), $\tilde{x}_i$ are the perturbations of the candidate passage x generated by a sampling model $q_\varphi(\tilde{x}|x)$, p_θ is the scoring model, $\tilde{\mu}$ and $\tilde{\sigma}$ are the sample mean and standard deviation of the log probabilities of the perturbations under p_θ, and the threshold ϵ is a tunable hyperparameter. The detector classifies x as machine-generated if its standardized log probability exceeds the threshold. eq. (1) is also valid for DetectGPT when $q_\varphi(\tilde{x}|x)$ is replaced by the perturbation distribution. The pseudo-code is presented in algorithm 1 for Fast-DetectGPT and algorithm 2 for DetectGPT.

2.3 Models

Fast-DetectGPT employs three distinct model components:

1. **Source model**: Generates candidate text by conditioning on human-written prefixes (first 30 tokens) and decoding from them.
2. **Sampling model** (q_φ): Provides conditional token distributions for generating perturbations $\tilde{x}_i$ of candidate passage x.
3. **Scoring model** (p_θ): Computes log-probabilities for evaluating perturbed sequences.

Detection experiments operate under two settings: *white-box* (identical source and scoring models) and *black-box* (real-life setting). The sampling model may differ from the scoring model in both settings, potentially improving detection performance when they are not identical.

2.4 Mid-K Attack

To evaluate detector robustness, we design the **Mid-K** perturbation attack. During generation, at a fixed number of randomly selected positions, the top-K most probable tokens in the model's next-token distribution are banned (their logits are set to $-\infty$), forcing the model to sample from less probable alternatives. Mid-K is controlled by three parameters: P, the number of positions per sequence where the ban is applied; K, the number of top tokens to exclude at each selected position; and S (optional), the first generated-token index eligible for perturbation (useful to avoid early tokens). For each sequence, we compute the maximum number of new tokens $M = \texttt{max\_length} - \texttt{prompt\_len}$ and select up to P distinct positions from the integer range $\{S, \dots, M\}$. During decoding, whenever the current new-token index t matches a selected position, we identify the top K token indices in the logits and set those logits to $-\infty$, then continue sampling from the remaining distribution.

3 Experiments

The experiments consist of comparing Fast-DetectGPT with three of the baselines referenced in the original study: Likelihood, Entropy and LogRank. The initial plan was to also compare them with DetectGPT, however, due to computational and time constraints, it could only be used one time (see Table 1). These methods are analyzed in various settings such as different model architectures, datasets and under the mid-k attack, which will be detailed below. All experiments are performed exclusively in the whitebox mode, where the source, sampling and scoring model is the same over the entire run. Performance is measured by the ROC AUC score.

3.1 Implementation

The implementation of our methods is based on the original code provided by the authors of Fast-DetectGPT [1]. We introduced all necessary modifications to adapt it to our new introduced models, datasets, Mid-K adversarial attack.

3.2 Reproducibility Study

The first step in our experiment is to evaluate the detection methods on different model architectures and datasets. We chose three small open-weight models of varying sizes: DeepSeek R1 Distill Llama 8B (DeepSeek-AI, 2025), Microsoft Phi 2 (2.7B parameters) (Javaheripi et al., 2023) and Mistral 7B Instruct v0.2 (Jiang et al., 2023). As for datasets, we compare a dataset used in the original study (XSum) with a newly introduced one (HC3). The aim is to understand if Fast-DetectGPT is robust to the topic, size and format of texts in its dataset, and to the model architecture and size, thus being reproducible.

3.3 Closed-Weight API Model

In addition to the three open-weight models, we investigate the performance of Fast-DetectGPT with a closed-weight API model, namely GPT 4o mini. Unlike the first three, with this model only the first 20 probabilities can be accessed. Therefore, it cannot be compared side-by-side with the open weight models, as that may be unfair. Still, the goal is to see if this limitation affects model performance.

[1] https://github.com/baoguangsheng/fast-detect-gpt

3.4 Low-Resource Language

Another experiment consists of evaluating Fast-DetectGPT on a dataset of a low resource language. For this we select the dataset XLSum (Hasan et al., 2021), an extension of XSum for various different languages. We specifically focus on the Scottish Gaelic subset, with the intention to observe robustness to language.

3.5 Adversarial Attack

Finally, we evaluate the robustness of the detectors under the **Mid-K** attack described. The goal is to measure how detection performance degrades as the attack strength increases.

Setup. We conduct all adversarial experiments using the **GPT-2** model on the **XSum** dataset. For each example, the generator is prompted with the first 30 tokens and sampled up to a maximum length of 200 tokens. Mid-K is applied during decoding, while the detection pipeline (Fast-DetectGPT) remains unchanged so that differences in performance reflect the impact of the attack alone.

We explored the following parameter grid: $P \in \{1, 5, 10, 20, 30, 50, 60, 70, 80, 90, 100, 150\}$, $K \in \{1, 2, 5, 10, 20\}$, and fixed $S = 5$.

Each grid point (P, K) corresponds to one experiment where Mid-K is applied at up to P positions starting from token index S.

Scope. Since the aim was to study the effect of Mid-K parameters rather than model or dataset variation, and given limited computational resources, we fixed $S = 5$ and restricted experiments to this single model–dataset pair. In total, we evaluated 60 parameter combinations covering a wide range of perturbation strengths.

4 Results

4.1 Reproducibility

The results for the main reproducibility experiment are reported in table 1. As expected, FastDetect-GPT displays the best performance over all models and datasets. The Likelihood and LogRank baselines are the next in the ranking, following Fast-DetectGPT closely with XSum but being sligthly lower with HC3, while Entropy visibly falls last. As mentioned before, DetectGPT could only be utilized one time, which was with R1-8B on XSum, its performance being only better than the Entropy baseline. This demonstrates the power of FastDetect-GPT over other methods.

Method	XSum			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	0.9999	0.8782	0.9665	0.9482
Entropy	0.1645	0.5411	0.4558	0.3871
LogRank	**1.0000**	0.9051	0.9596	0.9549
DetectGPT	0.8560	-	-	-
Fast-DetectGPT	**1.0000**	**0.9769**	**0.9989**	**0.9919**
(Diff)	0.1440	-	-	-

Method	HC3			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	0.8391	0.7139	0.7486	0.7672
Entropy	0.3521	0.4358	0.4235	0.4038
LogRank	0.8275	0.7224	0.7393	0.7631
Fast-DetectGPT	**0.9710**	**0.8347**	**0.8963**	**0.9007**

Table 1: Detection performance (AUROC) on XSum and HC3 datasets across three new source-models. Fast-DetectGPT is compared with three baselines, and only in 1 case with DetectGPT due to computational and time constraints. FastDetectGPT always achieves the best results.

Moreover, it can be noticed that the model size directly influences the score (i.e.: a higher model achieves better performance). However, the difference is not significant enough to prove a lack of robustness.

The results also differ from a dataset to another. It seems that most methods perform better on XSum than on HC3, while for Entropy it is the opposite case. The difference seems larger than that between models. This could be caused by the dataset size or by the diversity of text samples. Still, it does not deny the fact that Fast-DetectGPT dominates over the other detection methods.

4.2 API Model

Method	GPT 4o mini		
	XSum	**HC3**	**Avg.**
Likelihood	0.0001	0.0000	0.0000
Entropy	**0.9904**	**0.9985**	0.9945
LogRank	0.0000	0.0000	0.0000
Fast-DetectGPT	0.2065	0.2123	0.2094

Table 2: AUROC results for the API experiment with GPT 4o mini over the two datasets. This model shows strange behaviors, with the entropy baseline performing very well, while Fast-DetetGPT and the other baselines perform very poorly.

Table 2 below shows the GPT 4o mini results. This time, the model is characterized by unstable behaviors, leading to Fast-DetectGPT to perform rather poorly, and for two of the baselines to fail completely. Surprisingly, the Entropy baseline yields the best ROC AUC scores. This can be explained by ...

4.3 Low-resource Language

As for the Scottish-Gaelic language experiment, results can be seen in Table 3. This table demonstrates that all methods, with the exception of the Entropy baseline, keep a mostly consistent behavior and good, or even slightly better performance than on the English datasets. The Entropy however has a significantly lower performance than usual.

4.4 Mid-K Attack

Method	Underrepresented language			
	R1-8B	**Phi-2**	**Mistral**	**Avg.**
Likelihood	**1.0000**	0.9976	0.9972	0.9983
Entropy	0.0010	0.0968	0.2730	0.1236
LogRank	**1.0000**	**0.9984**	0.9974	**0.9986**
Fast-DetectGPT	0.9994	0.9825	**0.9995**	0.9938

Table 3: AUROC results for the under-represented language experiment. Surprisingly, Fast-DetectGPT performs slightly worse than the LogRank baseline.

We report results for the Mid-K adversarial attack applied to GPT-2 on the XSum dataset. Detection performance is measured by ROC AUC, and text quality by mean sampled perplexity.

Detection performance. Figure 2 shows that ROC AUC steadily decreases as the number of perturbed positions (P) increases. The unperturbed baseline achieves a ROC AUC of 0.9941, confirming that the detector performs almost perfectly without attacks. However, even moderate perturbations strongly degrade performance—for example, with $P = 50, K = 3$ the ROC AUC drops to 0.5287, and with $P = 60, K = 2$ it falls further to 0.512. In these cases, the attack causes little visible change to the text, yet detection becomes nearly random, indicating that the detector's signal is highly sensitive to subtle sampling perturbations.

Perplexity. As shown in Figure 1, mean perplexity grows with stronger perturbations

Overall, the Mid-K attack demonstrates that small, probabilistic manipulations of the decoding process can sharply reduce detector accuracy

5 Conclusion

Write conclusion here

A Additional Results

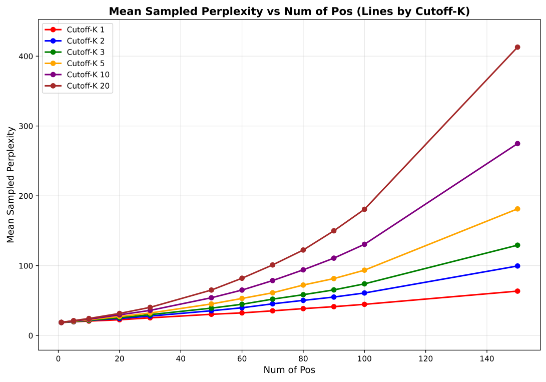

Figure 1: Mean sampled perplexity vs number of perturbations made. Presented per each cutoff tested. The trend follows inverse curve to the model performance. Measured on facebook/opt-125m (Zhang et al., 2022).

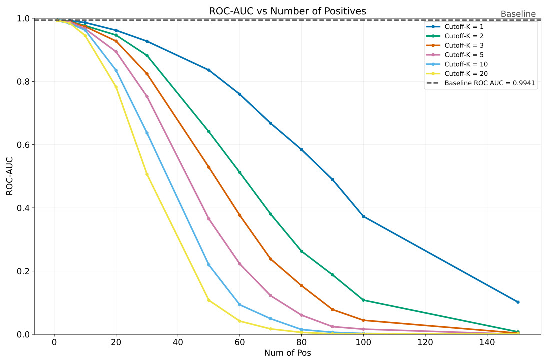

Figure 2: ROC AUC vs number of perturbed positions, shown separately for different cutoff values. The dashed line indicates baseline performance without perturbation. ROC AUC decreases as the number of perturbations increases, with larger cutoffs accelerating the degradation. Results are reported for GPT-2 (124M) (Radford et al., 2019).

Table 4: Example texts under Mid-K attack with GPT-2 (124M) on XSum.

Perturbation Results:

Pos / Limit	**Text**
Human	“It was an oversight,” Palmer told BBC Sport.
No perturbation	“It was an honest and open challenge over the length of time,” he said.
50 / 5	“It was an honest and open challenge over the length of time,” he tells us.
150 / 1	“It was an honest and open challenge over exactly how this particular scheme could have become so costly to operate after it closed in 2001, we still do to this present day,” said the report said
150 / 20	“It was an inappropriate response by the HL [Herodoplyosis Unit to complain that player Eddie Gabboni ran red before half goal]; in such situations players wear distinctive black shorts (most usually found throughout tournaments by Uprod on other social programs at venues as in RTS).

B Algorithms

Algorithm 1 Fast-DetectGPT machine-generated text detection.

Input: passage x, sampling model q_φ, scoring model p_θ, and decision threshold ϵ
Output: True – probably machine-generated, False – probably human-written.

```
1: function FastDetectGPT($x, q_\varphi, p_\theta$)
2:     $\tilde{x}_i \sim q_\varphi(\tilde{x}|x), i \in [1..N]$                      ▷ Conditional sampling
3:     $\tilde{\mu} \leftarrow \frac{1}{N}\sum_i \log p_\theta(\tilde{x}_i|x)$                       ▷ Estimate the mean
4:     $\tilde{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_i(\log p_\theta(\tilde{x}_i|x) - \tilde{\mu})^2$          ▷ Estimate the variance
5:     $\hat{\mathrm{d}}_x \leftarrow (\log p_\theta(x) - \tilde{\mu})/\tilde{\sigma}$      ▷ Estimate conditional probability curvature
6:     return $\hat{\mathrm{d}}_x > \epsilon$                                 ▷ Compare with threshold
7: end function
```

Algorithm 2 DetectGPT machine-generated text detection

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

References

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. 2019. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection. *Preprint*, arXiv:1907.09177.

Alim Al Ayub Ahmed, Ayman Aljabouh, Praveen Kumar Donepudi, and Myung Suh Choi. 2021. Detecting fake news using machine learning : A systematic literature review. *Preprint*, arXiv:2102.04458.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *Preprint*, arXiv:2301.07597.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3637–3647. ACM.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature.

Sandra Mitrović, Davide Andreoletti, and Omran Ayoub. 2023. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *Preprint*, arXiv:2301.13852.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *Preprint*, arXiv:1808.08745.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Wajiha Shahid, Yiran Li, Dakota Staples, Gulshan Amin, Saqib Hakak, and Ali Ghorbani. 2022. Are you a cyborg, bot or human?—a survey on detecting fake news spreaders. *IEEE Access*, 10:27069–27083.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.