Detection of LLM-Generated Text: Evaluation of Fast-DetectGPT

Laurent Vong
laurent.vong@ensae.fr
ENSAE

Abstract

The rapid advancement of large language models (LLMs) such as GPT-2 has enabled the automatic generation of human-quality text. This capability raises concerns regarding misinformation, academic plagiarism, and content forgery, highlighting the importance of detecting synthetic texts. This paper evaluates the Fast-DetectGPT detection method on mixed datasets (human vs. GPT-2 generated texts). We use three diverse datasets (XSum, SQuAD, PubMedQA) to test the algorithm's robustness across different content types. Experimental results demonstrate that Fast-DetectGPT offers robust and efficient detection capabilities, with performance varying across text types. Our analysis reveals failure cases related to text length or style, as well as potential improvements for future work.

1 Introduction

Large language models (LLMs) have revolutionized automatic text generation. GPT-2 exemplifies this capability, producing fluent text on numerous topics. This proliferation of generated text presents challenges: online misinformation, academic dishonesty, content counterfeiting, etc. Consequently, developing methods to automatically determine whether text was written by a human or generated by an LLM is essential.

In this work, our objective is to experimentally evaluate the Fast-DetectGPT algorithm and identify its performance in recognizing texts produced by GPT-2 across different datasets. We adopt a rigorous experimental approach: utilizing mixed datasets (human vs. AI) across several domains, then applying Fast-DetectGPT and analyzing its performance.

2 State of the Art in AI-Generated Text Detection

Various approaches have been developed to detect texts generated by LLMs. These can be classified into three main categories: supervised methods, unsupervised (zero-shot) methods, and watermarking-based methods.

2.1 Supervised Methods

Supervised methods employ classifiers trained on datasets of human and generated texts to learn distinguishing features. For example, GPT-Sentinel is a detector that uses models based on RoBERTa and T5 trained on a large corpus of texts paraphrased by ChatGPT. This approach achieves excellent performance ($> 97\%$ accuracy) on texts generated by the model on which it was trained but shows limitations when generalizing to new models.

2.2 Unsupervised Methods (Zero-Shot)

Zero-shot methods do not require specific training and can therefore adapt more easily to new models.

One of the first approaches was GLTR, which analyzes token probability distributions to help us identify generated texts. DetectGPT later introduced an approach based on the curvature of the model's log-probability function. This method observes that texts generated by an LLM tend to occupy regions of negative curvature in its log-probability function. It uses text perturbation via another model (T5) to calculate this curvature but remains computationally expensive.

Fast-DetectGPT significantly improves on DetectGPT's approach by replacing the perturbation phase with more efficient sampling, using the concept of conditional probability curvature. This optimization accelerates detection by approximately 340× while enhancing detection performance by about 75% in relative terms.

2.3 Adversarial Learning and LLMs as Detectors

Recent advancements have explored two additional promising approaches: adversarial learning and using LLMs themselves as detectors.

Adversarial learning methods create robust detectors by training them against evasion attempts. For instance, RADAR jointly trains a paraphraser and a detector in an adversarial framework, where the paraphraser attempts to generate realistic content that evades detection, while the detector aims to identify these sophisticated evasions. This approach has shown significant improvement in detection performance, especially when dealing with paraphrased content designed to avoid detection.

Another innovative approach involves using LLMs themselves as detectors. Studies have demonstrated that LLMs can be effective at distinguishing their own outputs from human writing. For example, some research has explored using LLMs to detect machine-generated text through in-context learning with adversarially generated examples. These methods leverage the inherent understanding of text generation patterns within the models themselves, turning their capabilities toward detection purposes.

3 Fast-DetectGPT: Methodology

3.1 Theoretical Principle

Fast-DetectGPT is founded on the observation that texts generated by LLMs exhibit statistical patterns distinct from human-written texts. Specifically, the algorithm introduces the concept of conditional probability curvature to characterize differences in word choices between LLMs and humans in a given context.

The method is based on the hypothesis that machines and humans make different choices during text generation. Machines are biased toward high-probability tokens due to their pre-training on large human text corpus, while individual humans do not exhibit this bias, as they construct sentences based on personal intention and context rather than statistics.

Formally, given a passage x and a potential source model p_θ, we quantify the conditional probability curvature as:

$$d(x, p_\theta, q_\phi) = \frac{\log p_\theta(x|x) - \mu_{\tilde{x}}}{\sigma_{\tilde{x}}} \tag{1}$$

where

$$\mu_{\tilde{x}} = \mathbb{E}_{\tilde{x} \sim q_\phi(\tilde{x}|x)}[\log p_\theta(\tilde{x}|x)] \tag{2}$$

and

$$\sigma_{\tilde{x}}^2 = \mathbb{E}_{\tilde{x} \sim q_\phi(\tilde{x}|x)}[(\log p_\theta(\tilde{x}|x) - \mu_{\tilde{x}})^2] \tag{3}$$

$\mu_{\tilde{x}}$ represents the expected score of samples $\tilde{x}$ generated by the sampling model $q_\phi(\cdot|x)$, and $\sigma_{\tilde{x}}^2$ the expected variance of the scores. In practice, we approximate $\mu_{\tilde{x}}$ using the average log-probability of random samples, and $\sigma_{\tilde{x}}^2$ using the mean of sample variances.

The conditional probability function is defined as:

$$p_\theta(\tilde{x}|x) = \prod_j p_\theta(\tilde{x}_j|x_{<j}) \tag{4}$$

where tokens $\tilde{x}_j$ are predicted independently given x.

The Fast-DetectGPT detector algorithm is presented below:

Algorithm 1 Fast-DetectGPT

```
**Require:** passage $x$, sampling model $q_\phi$, scoring model $p_\theta$, decision threshold $\epsilon$
**Ensure:** True - probably machine-generated, False - probably human-written
  $\tilde{x}_i \sim q_\phi(\tilde{x}|x), i \in [1..N]$ {Conditional sampling}
  $\mu_{\tilde{x}} \leftarrow \frac{1}{N}\sum_i \log p_\theta(\tilde{x}_i|x)$ {Estimate the mean}
  $\sigma^2_{\tilde{x}} \leftarrow \frac{1}{N-1}\sum_i(\log p_\theta(\tilde{x}_i|x) - \mu_{\tilde{x}})^2$ {Estimate the variance}
  $\hat{d}_x \leftarrow (\log p_\theta(x) - \mu_{\tilde{x}})/\sigma_{\tilde{x}}$ {Estimate conditional probability curvature}
  **return** $\hat{d}_x > \epsilon$
```

3.2 Implementation

Our Fast-DetectGPT implementation rigorously follows Algorithm 1 as defined above. The main functions of our code are:

- `sample_from_model(model, tokenizer, text, n_samples=100, temperature=1.0)`: Generates conditional samples by randomly replacing tokens at different positions. This function is essential to Fast-DetectGPT's efficiency.
- `compute_log_probs(model, tokenizer, text)`: Calculates token log-probabilities for a given text.
- `compute_curvature(model, tokenizer, text, n_samples=100)`: Calculates the normalized conditional probability curvature following Algorithm 1 exactly.
- `fast_detectgpt(model_name, datasets_path, n_samples=100, threshold=0)`: Orchestrates the detection process across a set of texts.

For implementation, we use PyTorch and the Hugging Face Transformers library to load and manipulate GPT-2 models. Independent conditional sampling is key to Fast-DetectGPT's efficiency. Specifically, we sample each token $\tilde{x}_j$ from $q_\phi(\tilde{x}_j|x_{<j})$ given the fixed passage x without depending on other sampled tokens.

4 Datasets

To rigorously evaluate Fast-DetectGPT, we used three different datasets covering a variety of styles and domains:

- **XSum**: A dataset of news article summaries, characterized by concise, factual style
- **SQuAD**: A question-answering dataset with information-rich contexts
- **PubMedQA**: A medical question dataset containing technical, specialized vocabulary

For each dataset, we:

- Selected 150 human texts
- Generated equivalent texts with GPT-2
- Limited text size to 1000 tokens maximum to avoid computation issues

5 Experiments and Results

5.1 Experimental Protocol

To evaluate Fast-DetectGPT's performance, we:

- Used 500 samples per text for curvature calculation
- Tested different classification thresholds
- Calculated several evaluation metrics: precision, recall, F1-score, and ROC AUC
- Analyzed curvature distributions for human and generated texts
- Visualized results with confusion matrices and ROC curves

5.2 Quantitative Results

Fast-DetectGPT's performance varies across datasets, as shown in Table 1.

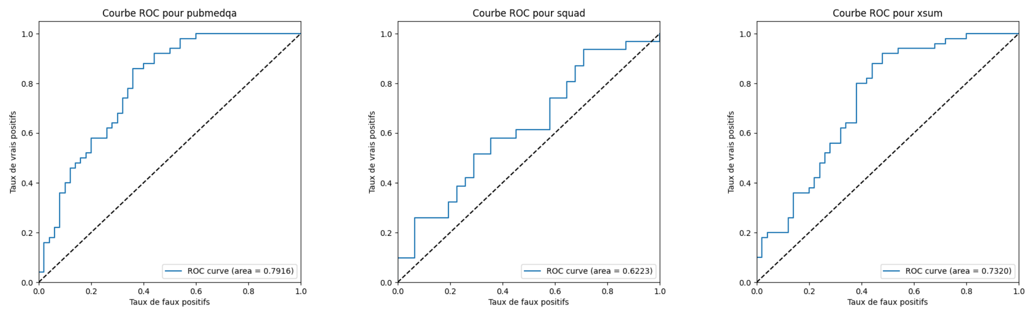
Table 1: Fast-DetectGPT performance on different datasets

Dataset	ROC AUC	Precision	Recall	F1-score
XSum	0.73	0.65	0.92	0.76
SQuAD	0.62	0.57	0.94	0.71
PubMedQA	0.79	0.68	0.92	0.78

These results show variable performance across datasets. The model achieves its best performance on PubMedQA, potentially due to the specialized and technical nature of medical texts presenting more distinctive patterns. Performance is lower on SQuAD, suggesting that question-answering contexts pose particular challenges for detection.

5.3 ROC and Precision-Recall Curve Analysis

Figure 1 presents the ROC curves for each dataset, illustrating the relationship between true positive rate and false positive rate at different decision thresholds.

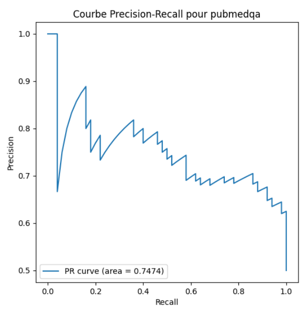


(a) PubMedQA (b) SQuAD (c) XSum

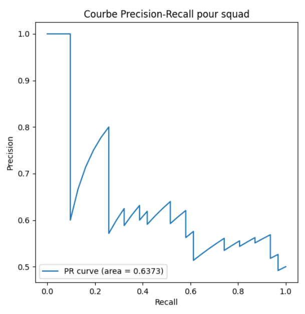
Figure 1: ROC curves for the three datasets

We also analyzed Precision-Recall curves (Figure 2), which show how precision and recall evolve as the decision threshold changes.

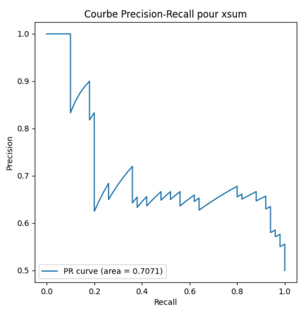
These curves reveal that Fast-DetectGPT maintains good precision at moderate recall levels, but precision decreases when recall exceeds 60-80%, especially for SQuAD. The area under the Precision-Recall curve (PR AUC) ranges from 0.64 for SQuAD to 0.75 for PubMedQA, confirming the performance hierarchy observed in ROC AUC scores.



(a) PubMedQA



(b) SQuAD

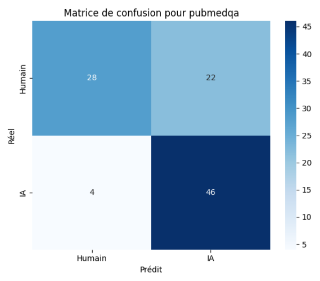


(c) XSum

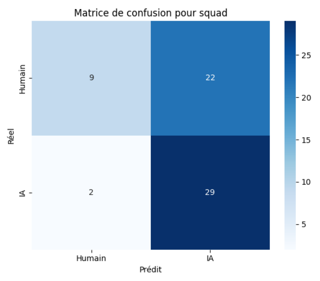
Figure 2: Precision-Recall curves for the three datasets

5.4 Confusion Matrix Analysis

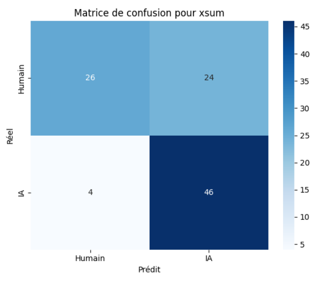
Confusion matrices (Figure 3) provide a detailed view of correct and incorrect predictions for each dataset.



(a) PubMedQA



(b) SQuAD



(c) XSum

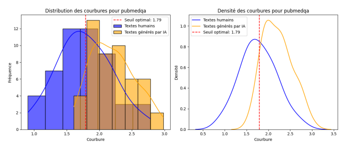
Figure 3: Confusion matrices for the three datasets

These matrices reveal a common pattern: Fast-DetectGPT is highly effective at detecting AI-generated texts (few false negatives) but has a relatively high rate of false positives (human texts classified as AI-generated). For PubMedQA, of 50 human texts, 28 are correctly identified while 22 are misclassified as AI. For SQuAD, only 9 human texts out of 31 are correctly identified. For XSum, 26 human texts out of 50 are correctly identified.

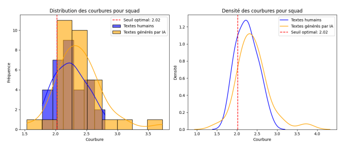
This tendency may be explained by the choice of a relatively low decision threshold that prioritizes AI-generated text detection at the expense of higher precision on human texts.

5.5 Curvature Distribution Analysis

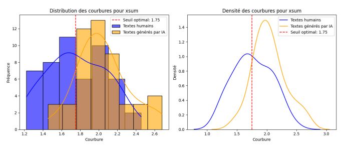
The analysis of curvature distributions (Figure 4) shows a separation between human and AI-generated texts, but with significant overlap:



(a) PubMedQA



(b) SQuAD



(c) XSum

Figure 4: Curvature distributions for the three datasets

For PubMedQA, human texts primarily show curvatures between 1.0 and 2.0, while AI-generated texts show curvatures between 1.8 and 3.0, with an optimal threshold around 1.79. For SQuAD, the separation is less distinct, with human curvatures between 1.7 and 2.5 and AI curvatures between 1.8 and 3.5, with an optimal threshold around 2.02. For XSum, human texts show curvatures between 1.2 and 2.2, and AI texts between 1.6 and 2.6, with an optimal threshold at 1.75.

These distributions show that overlap is greater for SQuAD, explaining the lower performance on this dataset. The best separation is observed for PubMedQA, consistent with the better performance obtained on this dataset.

Compared to the original paper's experimental setup, we used GPT-2 and only 500 samples instead of the 10,000 samples recommended in the implementation. This difference in methodology might partially explain the discrepancy between our results and those reported in the paper. Nevertheless, our findings align with the fundamental conclusion of the original research: human-written texts consistently exhibit lower conditional probability curvatures compared to LLM-generated texts, confirming the core hypothesis of the Fast-DetectGPT approach.

5.6 Performance-Influencing Factors

Several factors affect Fast-DetectGPT's reliability:

- **Text length**: Fast-DetectGPT is less reliable on very short texts (<100 tokens)
- **Lexical complexity**: Human texts with simple vocabulary may be incorrectly classified
- **Text style**: Technical or highly structured texts are more easily detectable
- **Sample count**: Increasing sample count improves accuracy but increases computation time

6 Error Analysis

Analysis of failure cases reveals several interesting patterns:

6.1 False Positives (Human Texts Classified as AI)

Human texts most often misclassified present the following characteristics:

- Low lexical diversity
- Highly factual or formal content following repetitive structures
- Extremely short texts where statistics are unreliable

6.2 False Negatives (Undetected AI Texts)

Generated texts that escape detection are often:

- Particularly creative or with atypical style
- Generations that produced patterns rarely observed in GPT-2 training
- Cases where curvature is near the decision threshold

7 Discussion

Our experiments show that Fast-DetectGPT exhibits variable performance across text types but generally provides good capability to identify AI-generated texts. The results suggest several important observations:

- **Domain influence**: Performance is better on medical texts (PubMedQA), likely because technical and specialized language presents more distinctive and predictable patterns.
- **Precision-recall trade-off**: Fast-DetectGPT achieves high recall (>90%) across all datasets, but at the cost of moderate precision (57-68%). This trade-off can be adjusted by modifying the decision threshold according to specific application needs.

- **Threshold sensitivity**: Curvature distribution analysis shows that threshold choice is crucial and depends on the dataset. A single optimal threshold for all text types does not exist.
- **Zero-shot method advantage**: Despite its limitations, Fast-DetectGPT offers the significant advantage of not requiring training on labeled examples, unlike supervised approaches that are more sensitive to domain changes.

These results suggest that Fast-DetectGPT is particularly well-suited to use cases where AI-generated text detection is prioritized, even at the cost of some false positives.

8 Conclusion and Future Directions

Our implementation and evaluation of Fast-DetectGPT confirms its effectiveness for detecting GPT-2 generated texts across different content types. With variable performance across datasets (ROC AUC from 0.62 to 0.79), this method offers an interesting compromise: it requires no supervised training while providing robust performance.

Fast-DetectGPT's main strengths are:

- Its zero-shot approach that requires no labeled data
- Its ability to work across different text types
- Its solid theoretical foundation based on LLM statistical properties

However, certain limitations persist:

- Reduced performance on very short texts
- High false positive rate on some datasets
- Sensitivity to decision threshold choice

For future work, several directions are possible:

- Dynamic threshold adaptation according to content type
- Combination with other detection methods to improve robustness

Our implementation is publicly available to enable other researchers to reproduce our experiments and extend this method to other models and datasets.

References

[1] Bao, G., Zhao, Y., Teng, Z., Yang, L., & Zhang, Y. (2024). Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature.

[2] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., & Finn, C. (2023). DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature.

[3] Gehrmann, S., Strobelt, H., & Rush, A. (2019). GLTR: Statistical Detection and Visualization of Generated Text.

[4] Chen, X., Cui, A., & Yang, D. (2023). GPT-Sentinel: Distinguishing Human and ChatGPT Generated Text.

[5] Wu, J., Yang, S., Zhan, R., Yuan, Y., Wong, D.F., & Chao, L.S. (2023). A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions.

[6] Hu, X., Chen, P. Y., & Ho, T. Y. (2023). RADAR: Robust AI-Text Detection via Adversarial Learning.

[7] Koike, R., Kaneko, M., & Okazaki, N. (2024). Outfox: LLM-generated essay detection through in-context learning with adversarially generated examples.