Detection of LLM Generation

DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature: Jiaxi Xiong

GPT-who: An Information Density-based Machine-Generated Text Detector: Ryan Zhang

A Watermark for Large Language Models: Yiding Yang

GPT-Sentinel: Distinguishing Human and ChatGPT Generated Content: Andrew Oh

DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature

Authors: Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, Chelsea Finn
Presented by: Jiaxi Xiong

Motivation & Introduction

Large Language Models (LLMs) like GPT-3, PaLM, and ChatGPT can produce text that is highly fluent and coherent.

Such models can be misused in education, journalism, and social media — e.g., AI-written essays or fake news.

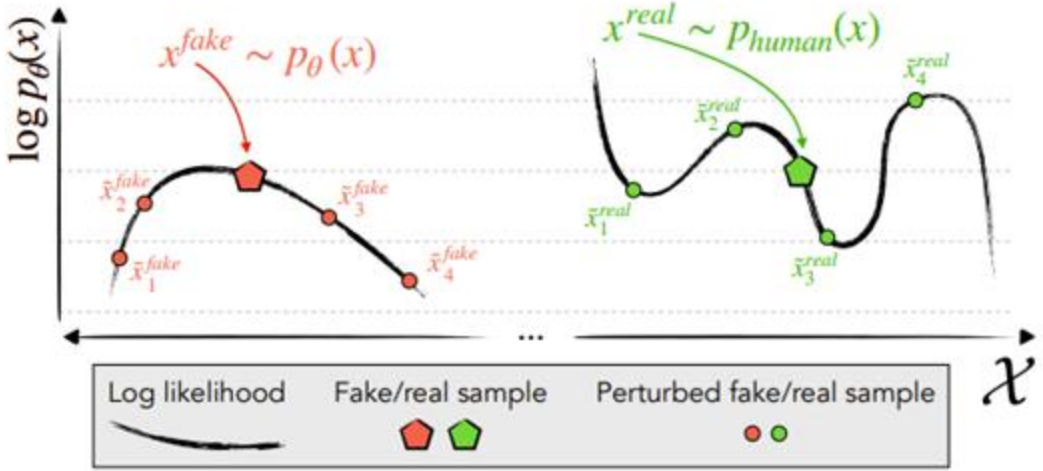
Humans perform only slightly better than random guessing when identifying AI-generated text.

Existing Methods & Limitation

- Classifier-based Approaches
  - Prior works train separate classifiers using labeled data
- Limitations
  - These classifiers often overfit to certain topics or models and must be retrained for each new LLM.
- Zero-shot Baselines are Simplistic
  - Simple zero-shot methods (like average log-probability) ignore local structure of the probability function.

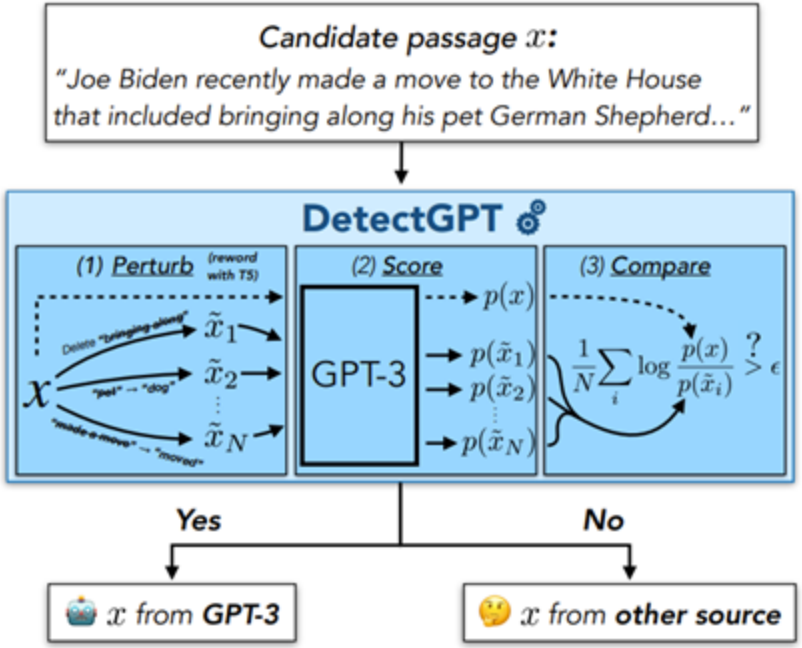
Key Hypothesis: Probability Curvature

- Negative Curvature for Model Text
- Human Text is More Stable
- DetectGPT Leverages This Property



DetectGPT Method Overview

- Perturb:Generate small paraphrases of the candidate text using a general model such as T5.
- Score: Compute the log-probabilities of the original and perturbed texts under the source model (e.g., GPT-3)
- Compare: If the original text's log-probability is much higher than the perturbed versions, it's likely machine-generated..



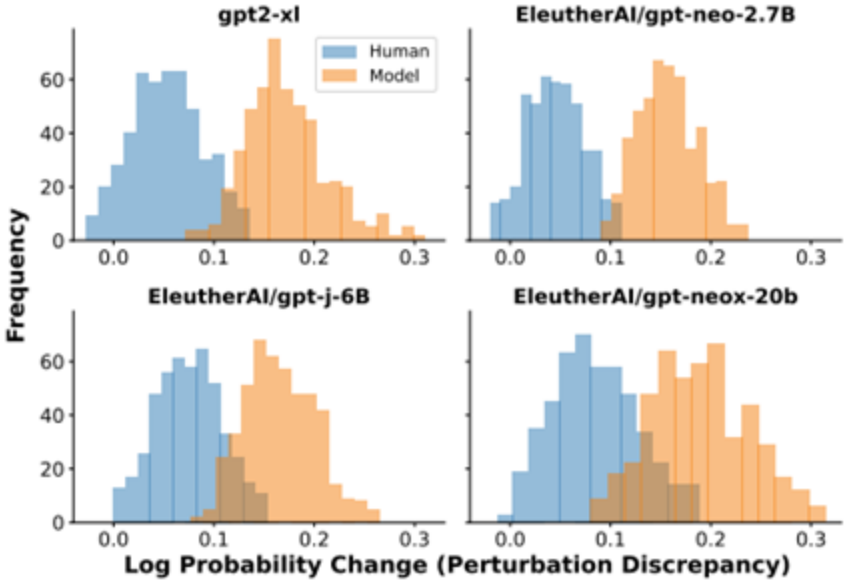
Algorithm1

Algorithm 1 DetectGPT model-generated text detection

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

Experimental validation of hypothesis

- Evaluate DetectGPT on various LLMs (GPT-2, GPT-J, GPT-Neo series) and datasets (XSum, SQuAD, WritingPrompts, PubMed).
- Use AUROC to measure how well the method separates human vs machine-generated texts.
- Model-generated texts show larger log-probability drops after perturbation than human texts → clear negative curvature.

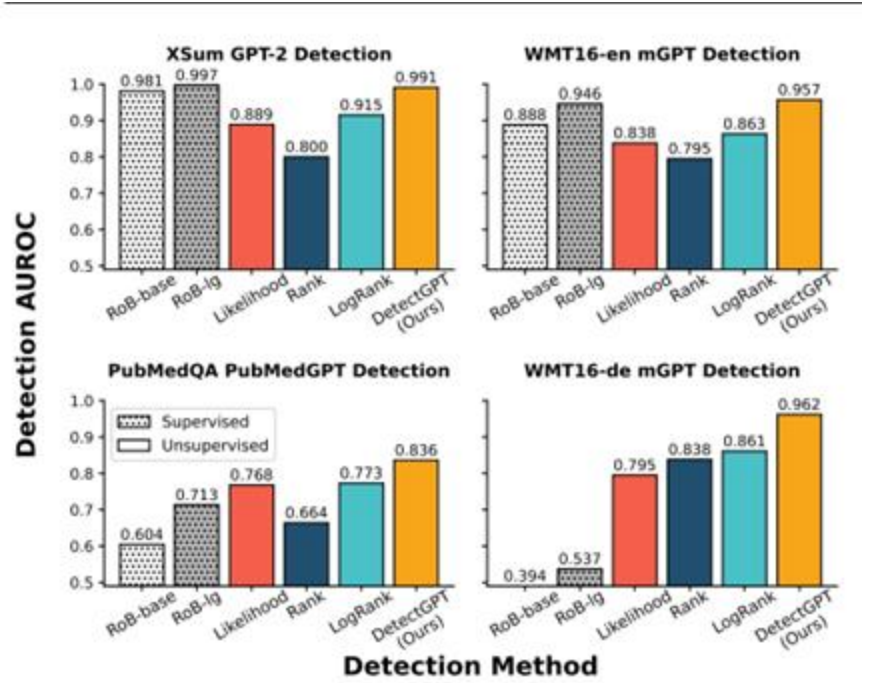


Quantitative results across models & datasets

	XSum						SQuAD						WritingPrompts					
Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	**Avg.**
$\log p(x)$	0.86	0.86	0.86	0.82	0.77	0.83	0.91	0.88	0.84	0.78	0.71	0.82	0.97	0.95	0.95	0.94	0.93*	0.95
Rank	0.79	0.76	0.77	0.75	0.73	0.76	0.83	0.82	0.80	0.79	0.74	0.80	0.87	0.83	0.82	0.83	0.81	0.83
LogRank	0.89*	0.88*	0.90*	0.86*	0.81*	0.87*	0.94*	0.92*	0.90*	0.83*	0.76*	0.87*	0.98*	0.96*	0.97*	0.96*	**0.95**	0.96*
Entropy	0.60	0.50	0.58	0.58	0.61	0.57	0.58	0.53	0.58	0.58	0.59	0.57	0.37	0.42	0.34	0.36	0.39	0.38
DetectGPT	**0.99**	**0.97**	**0.99**	**0.97**	**0.95**	**0.97**	**0.99**	**0.97**	**0.97**	**0.90**	**0.79**	**0.92**	**0.99**	**0.99**	**0.99**	**0.97**	0.93*	**0.97**
Diff	0.10	0.09	0.09	0.11	0.14	0.10	0.05	0.05	0.07	0.07	0.03	0.05	0.01	0.03	0.02	0.01	-0.02	0.01

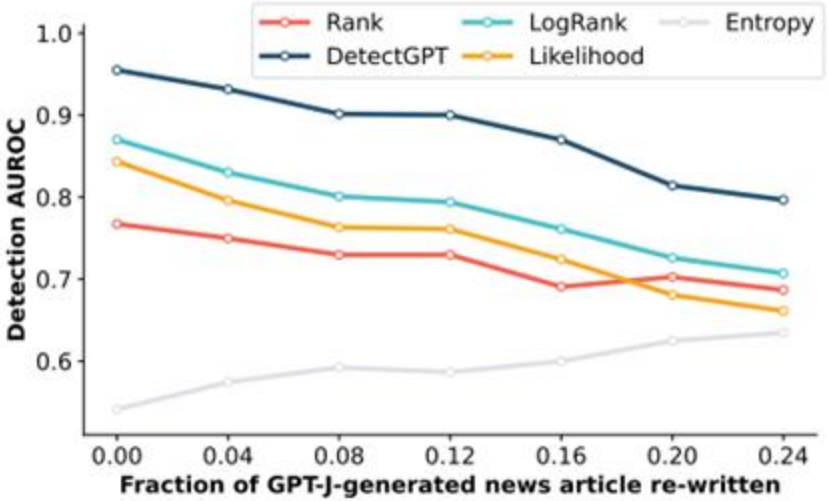
Comparison with Supervised Detectors

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

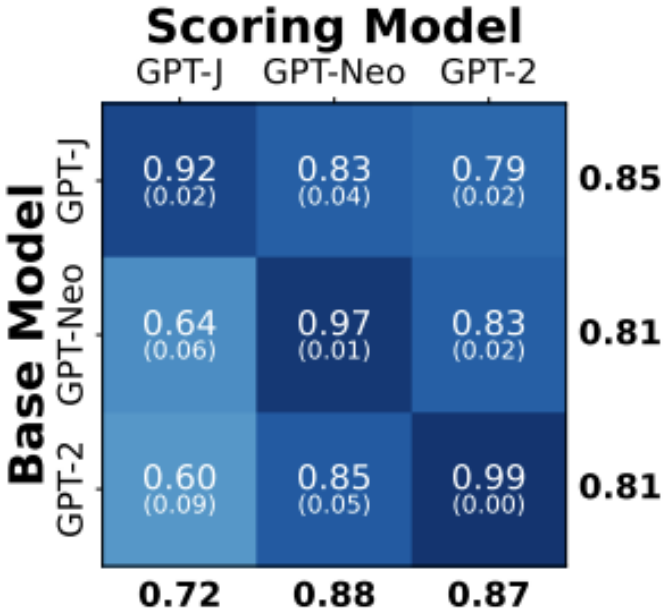


Comparison with Zero-Shot Baselines

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



Ablation Studies



DetectGPT performs best when scoring samples with the same model that generated them (diagonal).

But the column means suggest that some models (GPT-Neo, GPT 2) may be better ‘scorers’ than others (GPT-J).

Overall, this figure shows that DetectGPT performs best when it can directly access the generating model (**white-box**), but it remains fairly robust and effective even when the model is unknown (**black-box**).

Key Takeaways

Conceptual Breakthrough

DetectGPT leverages the negative curvature property of LLMs for zero-shot detection.

Practical Advantages

No training data needed, no classifier, works with any white-box LLM

Empirical Success

Outperforms zero-shot and supervised baselines across datasets and domains.

Future Directions

Address black-box models and improve efficiency via Fast-DetectGPT.

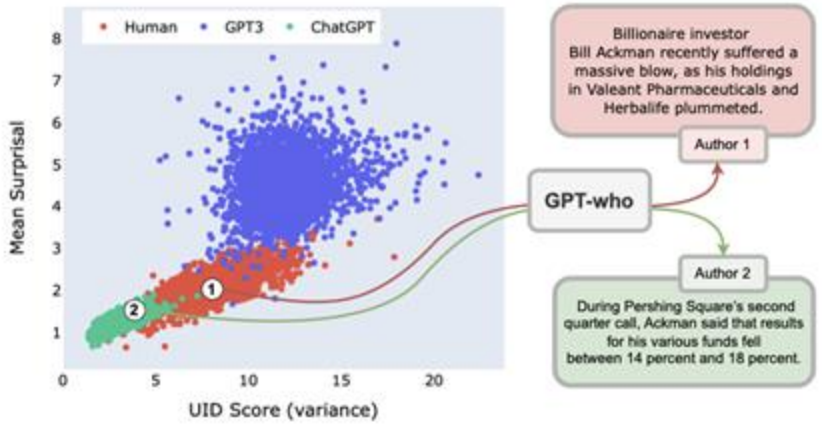
GPT-who: An Information Density-based Machine-Generated Text Detector

Authors: Saranya Venkatraman, Adaku Uchendu, Dongwon Lee

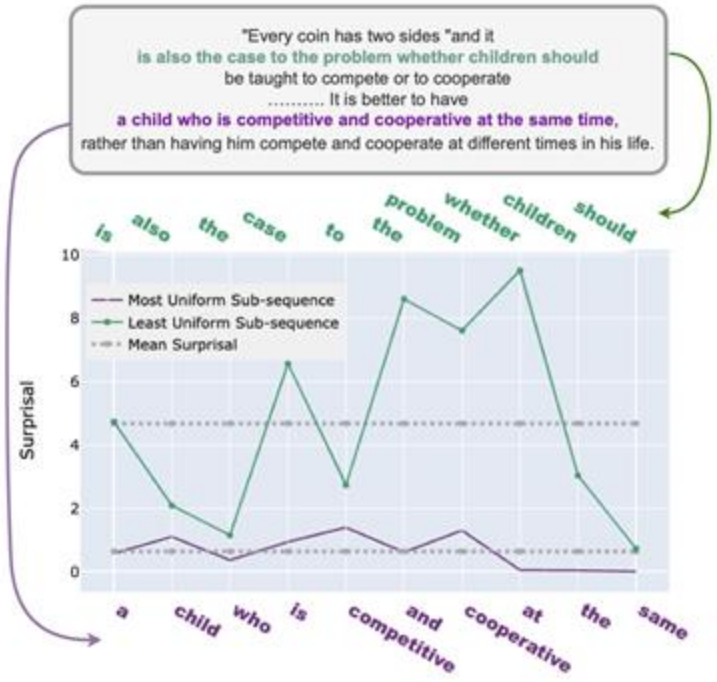
Presented by Ryan Zhang

Motivation & Introduction

- Explosion of LLM Generated Content: LLMs can write essays, news, and code indistinguishable from humans
- Detection is key to controlling misinformation, plagiarism, etc.
- Detection: Turing Test (TT) vs. Authorship Attribution (AA)
- GPT-who offers a simple solution based on Uniform Information Density (UID)

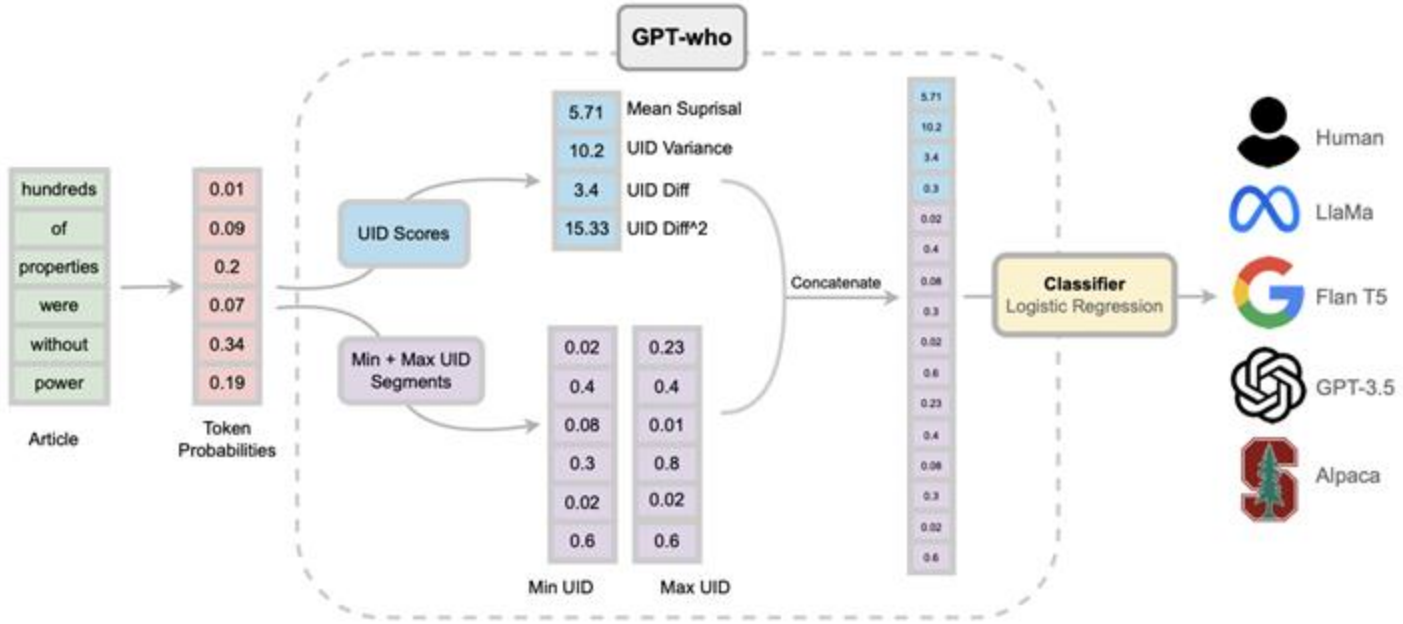


Background: UID



- Surprisal = “Unexpectedness” of a word in a given context
- High surprisal → rare / unpredictable word
- Low surprisal → common / expected word
- Humans distribute information evenly → stable surprisal

Method Pipeline



UID Feature Design

- **Mean surprisal (μ):** overall info level

$$\mu(y) = \frac{1}{|y|}\sum_{t}(u(y_t))$$

- **UID variance:** global smoothness

$$\mathrm{UID}(y) = \frac{1}{|y|}\sum_{t}(u(y_t) - \mu)^2$$

- **UID diff / diff²:** local changes

$$\mathrm{UID}(y) = \frac{1}{|y|-1}\sum_{t=2}^{|y|} abs(\mu(y_t) - \mu(y_{t-1}))$$

- **Min / Max UID spans (Newly Created Feature):** extreme low/high-info regions

Datasets & Baselines & Results

- **Datasets:** TuringBench, GPA-Bench, ArguGPT, In-the-Wild
- **Baselines:** GLTR, ZeroGPT, DetectGPT, OpenAI's detector, LongFormer-based detector, fine-tuned BERT

Task Type	Domain	GPTZero	ZeroGPT	OpenAI Detector	DetectGPT	BERT	ITW	GPT-who
Task 1	CS	0.30	0.67	0.81	0.58	**0.99**	0.98	**0.99**
	PHX	0.25	0.68	0.70	0.54	**0.99**	0.98	0.98
	HSS	0.72	0.92	0.63	0.57	**0.99**	0.96	0.98
Task 2	CS	0.17	0.25	0.64	0.16	**0.99**	0.81	0.84
	PHX	0.06	0.10	0.24	0.17	**0.96**	0.76	0.90
	HSS	0.44	0.62	0.27	0.20	**0.97**	0.29	0.80
Task 3	CS	0.02	0.03	0.06	0.03	**0.97**	0.38	0.63
	PHX	0.02	0.03	0.04	0.05	**0.97**	0.31	0.75
	HSS	0.20	0.25	0.06	0.06	**0.99**	0.08	0.62
Average F1		0.24	0.40	0.38	0.26	**0.98**	0.62	0.83

Results (continued)

Human v.	GROVER	GTLR	GPTZero	DetectGPT	RoBERTa	BERT	ITW	Stylometry	GPT-who
GPT-1	0.58	0.47	0.47	0.51	0.98	0.95	0.92	0.99	**1.00**
GPT-2_small	0.57	0.51	0.51	0.51	0.71	0.75	0.47	0.75	**0.88**
GPT-2_medium	0.56	0.49	0.50	0.52	0.75	0.65	0.47	0.72	**0.87**
GPT-2_large	0.55	0.46	0.49	0.51	0.79	0.73	0.46	0.72	**0.88**
GPT-2_xl	0.55	0.45	0.51	0.51	0.78	0.79	0.45	0.73	**0.89**
GPT-2_PyTorch	0.57	0.72	0.50	0.52	0.84	**0.99**	0.47	0.83	0.85
GPT-3	0.57	0.35	0.47	0.52	0.52	0.79	0.48	0.72	**0.84**
GROVER_base	0.58	0.39	0.52	0.51	**0.99**	0.98	0.49	0.76	0.81
GROVER_large	0.54	0.41	0.47	0.52	**0.99**	0.98	0.52	0.71	0.75
GROVER_mega	0.51	0.42	0.42	0.51	0.94	**0.97**	0.53	0.68	0.72
CTRL	0.49	0.88	0.67	0.67	**1.00**	**1.00**	0.91	0.99	0.99
XLM	0.50	0.89	0.67	0.67	0.58	**1.00**	0.92	0.96	0.99
XLNET_base	0.58	0.75	0.51	0.67	0.79	**0.99**	0.84	0.95	0.98
XLNET_large	0.58	0.88	0.67	0.52	**1.00**	**1.00**	0.93	**1.00**	**1.00**
FAIR_wmt19	0.56	0.56	0.56	0.51	0.84	**0.93**	0.49	0.74	0.74
Fair_wmt20	0.58	0.49	0.50	0.51	0.45	0.47	0.47	0.73	**1.00**
TRANSFO_XL	0.58	0.35	0.49	0.52	0.96	**0.97**	0.81	0.79	0.79
PPLM_distil	0.59	0.64	0.52	0.67	0.90	0.88	0.51	0.92	**0.95**
PPLM_gpt2	0.58	0.68	0.51	0.51	**0.90**	0.89	0.49	0.88	0.89
Average F1	0.56	0.57	0.52	0.55	**0.88**	0.61	**0.88**	0.82	**0.88**

Results (continued)

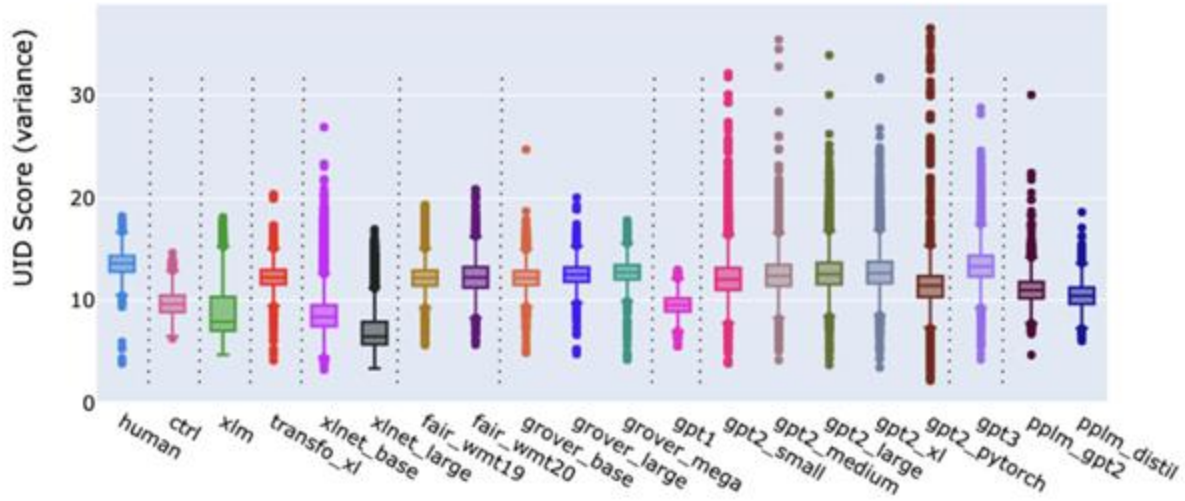
Detection Setting	Testbed Type	GPTZero	GLTR	DetectGPT	BERT	ITW	GPT-who
In-distribution	Domain-specific Model-specific	0.65	0.94	0.92	**0.98**	0.97	0.93
	Cross-domains Model-specific	0.63	0.84	0.6	**0.98**	0.97	0.88
	Domain-specific Cross-models	0.57	0.8	0.57	0.49	**0.87**	0.86
	Cross-domains Cross-models	0.57	0.74	0.57	0.49	0.78	**0.86**
Out-of-distribution	Unseen Models	0.58	0.65	0.6	**0.84**	0.79	0.74
	Unseen Domains	0.57	0.72	0.57	0.68	**0.8**	0.77
	Average F1	0.60	0.78	0.64	0.74	**0.86**	0.84

Author	Experts*	Stylometry	BERT	GPT-who
text-babbage-001	0.47	0.45	0.84	**0.85**
text-curie-001	0.47	0.45	0.83	**0.84**
text-davinci-003	0.66	0.59	**0.95**	0.77
gpt-3.5-turbo	0.63	0.69	**0.96**	0.84
gpt2-xl	0.37	0.49	**0.95**	0.91
Average F1	0.52	0.53	**0.91**	0.84

- Outperforms statistical detectors (GLTR, GPTZero, DetectGPT) by large margins
- Comparable to fine-tuned transformers on 2 of 4 benchmarks

UID Signatures of Authors

- Test whether UID patterns can distinguish humans from machines
- Examine if different LM families show unique UID “signatures”



Training & Inference Efficiency

- Measured training + inference time on 6 In-The-Wild (largest benchmark) testbeds
- GPT-who fastest — only one LM forward pass + logistic regression
- Competing methods are far slower due to LM fine-tuning (BERT) or multiple inference calls (DetectGPT) needed

Method	One-Time Training	Inference
DetectGPT	>10 hours	60 sec
BERT	~1.5 hours	2 sec
Stylometry	~1.5 hours	2 sec
GPT-who	20 min	0.8 sec

Discussion

- Human text appears more non-uniform than machine text.
- Does not contradict UID theory: uniformity is relative to each author’s probability distribution.
- Goal was not to prove UID theory but to test its utility for detection.
- UID-based features still distinguish humans vs machines and across LM families, independent of theory alignment.

Conclusion & Limitations

- GPT-who: UID-based, domain-agnostic statistical detector
- Outperforms other statistical methods and nears fine-tuned models
- More efficient with no training due to its psycholinguistic basis
- Limitations: limited datasets; needs tests on more tasks like QA or summarization

A Watermark for Large Language Models

Authors: John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, Tom Goldstein
University of Maryland

Presented by: Yiding Yang

Why watermark?

Problem Background:

- LLMs can automatically generate high-quality text.
- Write academic papers? Fabricate fake news? Forge content?

We hope that

- Could we have the model embed a hidden watermark within the text itself so that we can detect

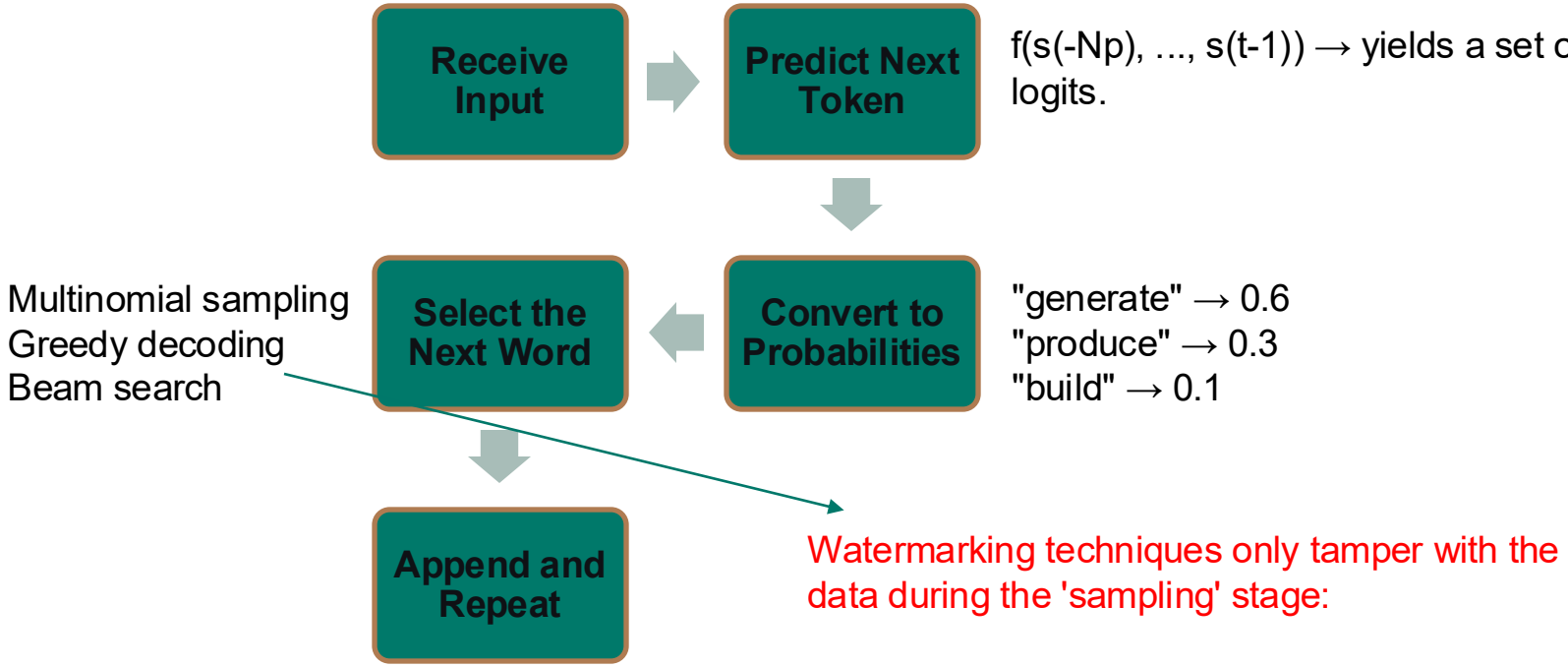
Our goals

- Detectable without access to model parameters or API
- No retraining required to generate watermarked text
- Detection possible from partial text
- Watermark is hard to remove
- Statistically rigorous detection with p-value

A language model is a function that predicts the probability of the next word.

Symbol	Meaning	Explanation
V	Vocabulary	The set of all available "tokens" (words or word pieces) for the model. Typically has **
T	Generated Token Sequence Length	The number of tokens in a generated piece of text.
s(t)	The t-th token	For example, s(1)="The", s(2)="model", etc.
s(-Np)...s(-1)	Prompt	The part input by the user, e.g., "Explain the theory of".
s(0)...s(T)	Model-generated Output Sequence	The content generated by the model based on the prompt.
f	Language Model Function	The prediction function of the model, parameterized by a neural network.
logits	Raw Output Score Vector	A vector of dimension
softmax	Converts logits to a probability distribution	Obtains the generation probability for each word.

How Autoregressive Language Models Generate Text



Entropy

High Entropy: Many plausible continuations for a sentence.

- "Today I feel ..."
- Possible next tokens: "happy", "sad", "tired", "motivated"...

High-entropy sentences →Easy to watermark, easy to detect.

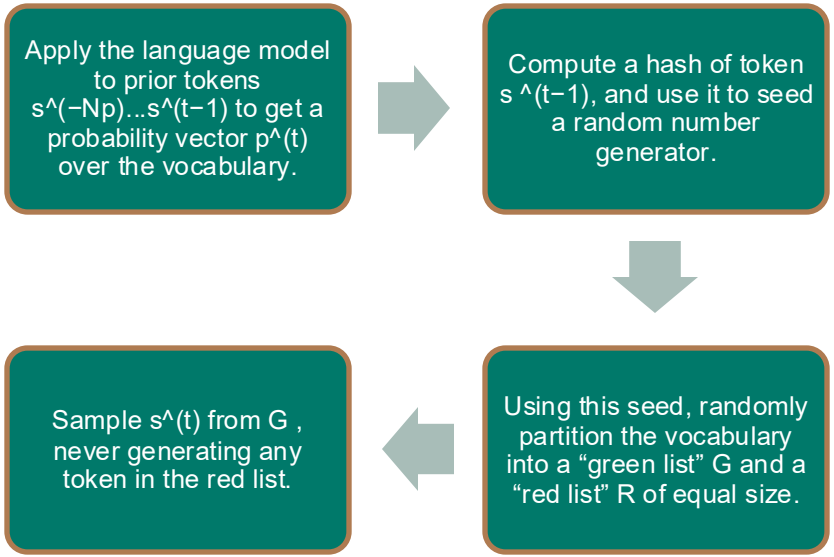
Low Entropy: Only one or a few highly probable continuations.

- "The capital of France is …"
- The almost certain next token: "Paris"

Low-entropy sentences → Difficult to watermark, challenging to detect.

Hard Red List

Process



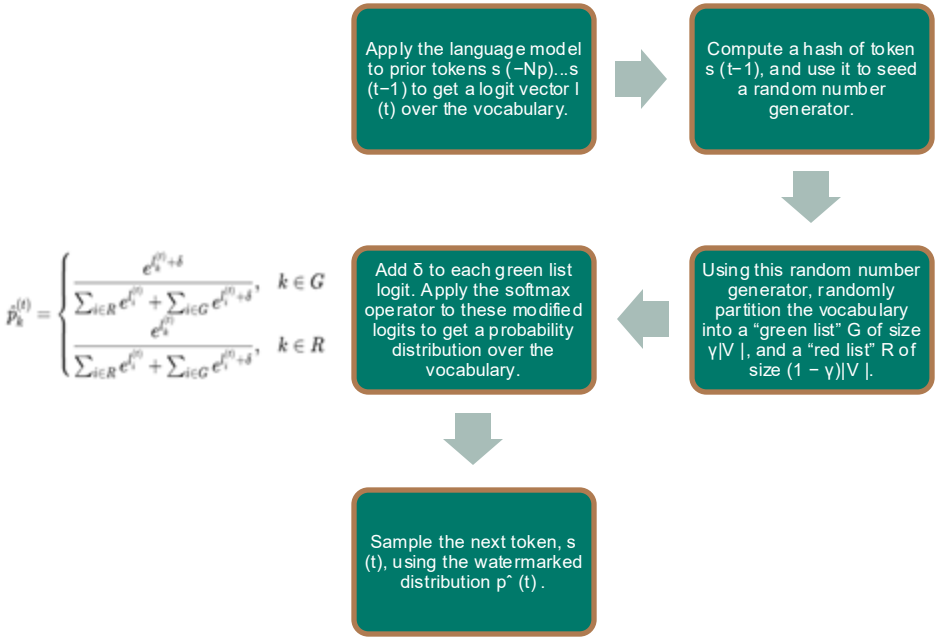
Detection Methods

- For each token, reconstruct the corresponding red/green list.
- Count the number of tokens falling into the green list within this text segment, denoted as $|s|_G$.
- Theoretically:
  - For natural text (unwatermarked): Each token has a 50% probability of falling into green → Expected value $E(|s|_G) = T/2$.
  - For watermarked text: 100% of tokens are in the green list.

$$z = \frac{2(|s|_G - T/2)}{\sqrt{T}}$$

Soft Watermark

Process



Detection Methods

- We assume the null hypothesis H_0: The text is naturally written, without watermarks.
- Then we calculate the z-statistic:

$$z = \frac{(|s|_G - \gamma T)}{\sqrt{T\gamma(1-\gamma)}}$$

- If the z-value is significantly greater than 0 (e.g. z>4), this indicates that the proportion of green words far exceeds that expected by chance.

Analysis of the Soft Watermark

Spike entropy

$$S(p,z) = \sum_k \frac{p_k}{1 + z p_k}$$

p is the probability distribution output by the model, z is a tuning parameter.

- When the distribution is highly 'peaked' (with most probability concentrated on a single word),S is small;
- When the distribution is 'smooth' (with probability distributed more evenly), S is large.

Assume that during token generation, the language model:
- applies a bias δ to green-listed words (i.e. α = e^δ),"
- with a green-list proportion of γ,
- an average spike entropy no less than $S*$

The number of green list words in the generated text $|s|_G$ satisfies:

$$\mathbb{E}[|s|_G] \geq \frac{\gamma\alpha T}{1+(\alpha-1)\gamma} S^*$$

$$\mathrm{Var}(|s|_G) \leq T \frac{\gamma\alpha S^*}{1+(\alpha-1)\gamma}\left(1 - \frac{\gamma\alpha S^*}{1+(\alpha-1)\gamma}\right)$$

When γ ≥ 0.5, a simpler upper bound may be employed: $\mathrm{Var}(|s|_G) \leq T\gamma(1-\gamma)$

- The higher the expected value (E) of the average green word count, the stronger the watermark and the easier it is to detect;
- The lower the variance (Var), the greater the stability of detection.

Analysis of the Soft Watermark

type-II error analysis to calculate the sensitivity

- $\gamma = 0.5$, $\delta = 2$
- Detection threshold $z = 4$ (corresponding to a false alarm rate of 3×10^{-5})
- Generation length $T = 200$ tokens
- Using the OPT-1.3B model
- Tested on the RealNewsLike subset of the C4 dataset

Experiment results

Metric	Value	Interpretation
Average spike entropy (S)	0.807	Text is fairly diverse
Expected green tokens (theory)	≥ 142.2	Predicted signal strength
Empirical mean	159.5	Watermark stronger in practice
Standard deviation (σ)	≤ 6.41	Stable across samples
Detection sensitivity	98.6%	Almost all detected
Empirical detection rate	98.4% (multinomial), 99.6% (beam search)	Very high accuracy

Private Watermark

pseudorandom function, PRF: FK (·)
The way to generate private watermark:

1. Select a random key K.

2. When generating each token, use the preceding h tokens as input to compute: $FK(s^{(t-h)},...,s^{(t-1)})$ to obtain a pseudorandom output.

3. This output determines which tokens belong to the green list and which belong to the red list.

Improve: Robust Private Watermarking

Input prompt, set PRF, key K, parameter δ, window h;

For each step, generate:

- Compute logits;
- Sort in descending order of probability;
- Provisionally select the current most probable word st and compute its hash;
- Pseudorandomly determine whether it belongs to the green or red list;
- If it falls into the red list and the logit gap is too large (below δ), skip it and select the next candidate;

If the threshold δ is exceeded, retain the red token and proceed to the next step.

Experiments

- Can watermarks be accurately detected?
- Does the watermark compromise the quality of the text?
- What is the effect of different parameters on detection intensity?

Set up

Data:

- "News-like" subset from C4 (Colossal Clean Crawled Corpus).
- Each sample comprises two parts: prompt (input) + completion

Generation method:

- Generated 200±5 tokens using different decoding strategies:
  - Multinomial sampling
  - Greedy decoding / Beam search (4 beams, 8 beams)
- To prevent premature <EOS> output during beam search, the EOS token was masked to ensure consistent segment lengths.

Test Metrics:

- z-score: Detection statistic
- Perplexity (PPL): Text fluency metric
- ROC curve, AUC: Detection performance
- Type-I error (false positive)
- Type-II error (false negative)

Results

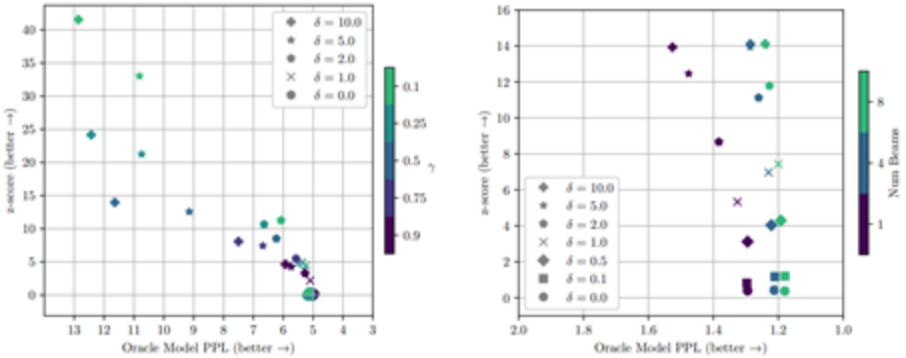


Figure 2 — Trade-off between watermark strength and text quality

Experiments

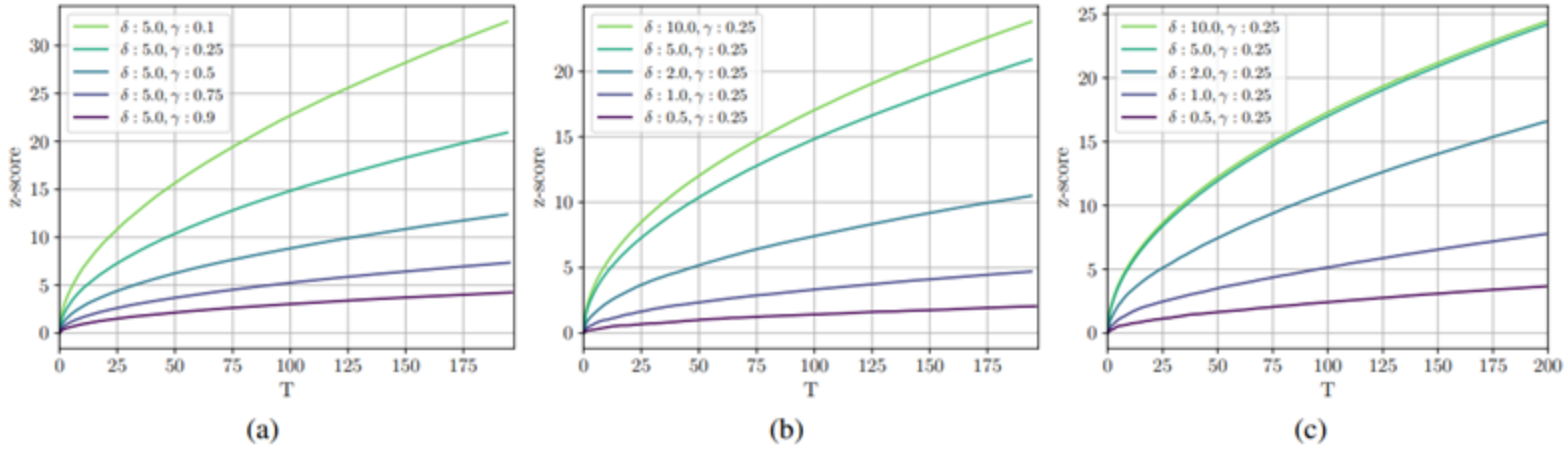


Figure 3 — Relationship between z-score and text length T

- The z-score increases monotonically with text length T.
- The theoretical prediction holds true: longer texts enable more stable detection of watermarks.
- Both δ and γ can be adjusted to regulate detection intensity.

Experiments

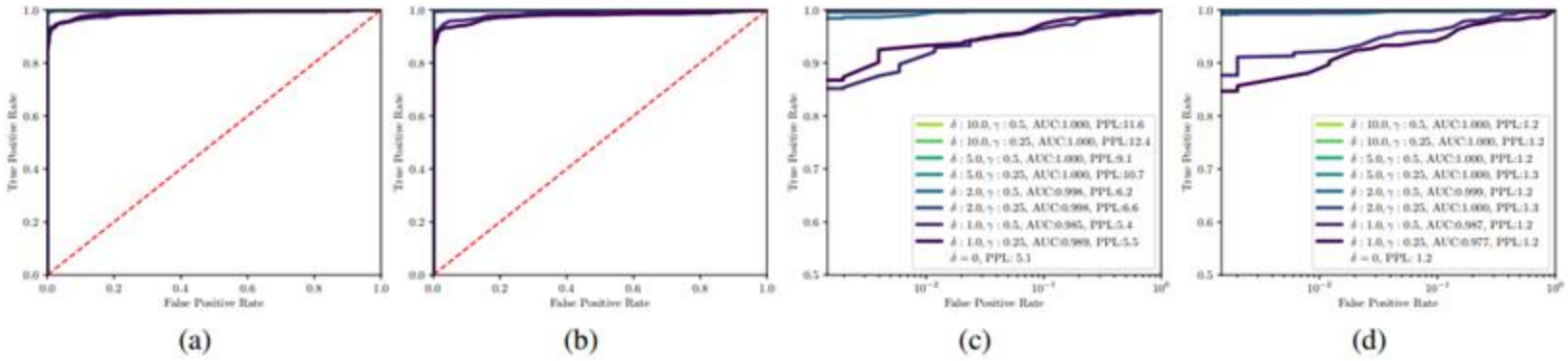


Figure 4 — ROC Curve and Detection Performance

- (a)(b) Linear coordinates, (c)(d) Logarithmic coordinates.
- When $\delta \geq 5$, the AUC approaches 1.0 → Detection is nearly perfect.
- Beam search again slightly outperforms random sampling.
- This indicates the statistical characteristics of the watermark signal are highly stable, making it resistant to misclassification or loss.

Experiments

				z=4.0				z=5.0			
sampling	δ	γ	count	FPR	TNR	TPR	FNR	FPR	TNR	TPR	FNR
m-nom.	1.0	0.50	506	0.0	1.0	0.767	0.233	0.0	1.0	0.504	0.496
m-nom.	1.0	0.25	506	0.0	1.0	0.729	0.271	0.0	1.0	0.482	0.518
m-nom.	2.0	0.50	507	0.0	1.0	0.984	0.016	0.0	1.0	0.978	0.022
m-nom.	2.0	0.25	505	0.0	1.0	0.994	0.006	0.0	1.0	0.988	0.012
m-nom.	5.0	0.50	504	0.0	1.0	0.996	0.004	0.0	1.0	0.992	0.008
m-nom.	5.0	0.25	503	0.0	1.0	1.000	0.000	0.0	1.0	0.998	0.002
8-beams	1.0	0.50	495	0.0	1.0	0.873	0.127	0.0	1.0	0.812	0.188
8-beams	1.0	0.25	496	0.0	1.0	0.819	0.181	0.0	1.0	0.770	0.230
8-beams	2.0	0.50	496	0.0	1.0	0.992	0.008	0.0	1.0	0.984	0.016
8-beams	2.0	0.25	496	0.0	1.0	0.994	0.006	0.0	1.0	0.990	0.010
8-beams	5.0	0.50	496	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
8-beams	5.0	0.25	496	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000

Table 2 — Experimental Error Rate

- All experiments: FPR=0.0 → No human text was misclassified.
- When δ=2, FNR ≤ 1.6% (i.e., false negative rate below 2%).
- When δ≥5, TPR=1.0, FNR=0.0 → Complete detection success.
- Beam search yields results nearly identical to multinomial models, and even slightly superior.

Attacking the Watermark

Three major categories of attack

Insertion Attack

- The attacker introduces additional tokens into the text, such as symbols or random words.

Deletion Attack

- The attacker removes several tokens, particularly words from the green list, thereby diminishing the watermark signal.

Substitution Attack

- Attackers rewrite words using synonyms or paraphrases (e.g., "good" → "great").

T5 Span Replacement Attack

Goal:

Simulate an attacker automatically replacing portions of the original text using a weaker language model (T5-Large) to undermine the watermark.

Experimental Setup:

- Attack target: A watermarked OPT-1.3B model output (γ=0.5, δ=2.0).
- Attacker model: T5-Large.
- Attack methodology:
  1. Tokenise the text;
  2. Randomly select tokens to replace with <mask>;
  3. Predict 20 candidates at that position using T5 (beam search=50);
  4. Overwrite with the new word if replacement succeeds;
  5. Continue until the replacement ratio reaches ε (attack budget), e.g. ε=0.1 indicates 10% modification.

Attacking the Watermark

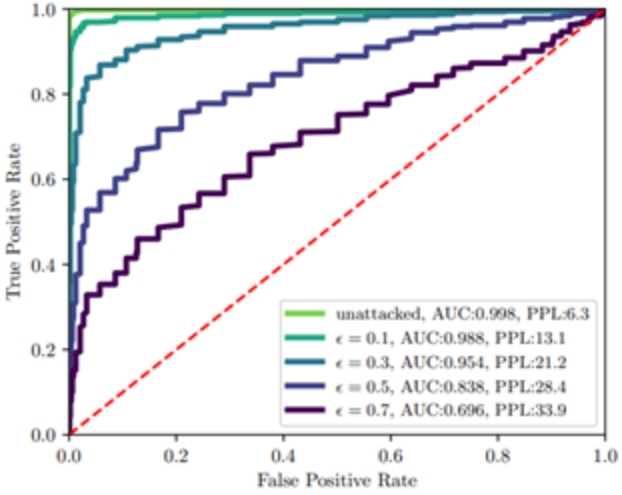


Figure 6. ROC curves for watermark detection under attack

Result Analysis:

- ε = 0.1 → Virtually ineffective, with AUC declining by merely 0.01 (detection remains robust).
- ε = 0.3 → Watermarking significantly weakened, yet text perplexity PPL surged to threefold the original value.
- ε ≥ 0.5 → Although watermarking is nearly eliminated, sentences become severely distorted with abysmal quality.

Conclusion:

To genuinely remove the watermark, an attacker must rewrite over 30% of the content, incurring an extremely high cost, with the generated text quality demonstrably deteriorating.

Conclusion

1. Summary of Key Contributions

- Simple and user-friendly
- Theoretically provable
- Virtually no false positives
- Strong resistance to attacks
- High portability

2.Technical Highlight: Independence of the z-statistic

The z-statistic employed during testing relies solely on γ (the size of the green list) and the hash function, and is independent of δ (the logit bias).

3.Scalable Deployment Approach

• Be enabled only in specific contexts (e.g., when user behaviour is deemed suspicious);
• Assign different δ values for distinct tasks;
• Be utilised at the model API level to automatically detect whether "suspected abusive generation" has occurred.

4.Open Questions (Future Work)

- 1. More robust hashing rules: Does a theoretically optimal hashing scheme exist?
- 2. Short text detection: Can reliable detection be achieved using only partial text fragments?
- 3. More precise sensitivity bounds under large δ / small γ: Theoretical improvements remain possible.

GPT-Sentinel: Distinguishing Human and ChatGPT Generated Content

Authors: Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, Bhiksha Raj (2023)

Present by Andrew Oh

Context

- Introduction & Motivation
- Related Work
- About Dataset
- RoBERTa-Sentinel
- T5-Sentinel
- Training and Evaluation
- Results
- Confidence
- Interpretability
- Future work & Conclusion

Motivation & Goal

- AI generated text is rapidly growing and threatens authenticity
- Existing detectors lack generalization and interpretability
- Need robust AI-Text detection framework for modern LLM
- Final Goal is to detect whether text is human or ChatGPT generated

Related Work

1. Accuracy -How well can it distinguish LLM text from human text(Precision, Recall)?
2. Data Efficiency- How well does it learn from small data?
3. Generalizability- Does it work on unseen models?
4. Interpretability- Can the model's decision be explained?

Statistical methods drop in accuracy as model size increases (88% → 74%)

- Small models(GPT-2 124M), accuracy = 88%
- Large models(GPT-2 1.5B), accuracy = 74%

GLTR improves detection (54% → 72%) via token probability analysis but lacks learning.

Zero-shot approaches underperform, while fine-tuned LMs (e.g., RoBERTa) achieve ~90% accuracy.

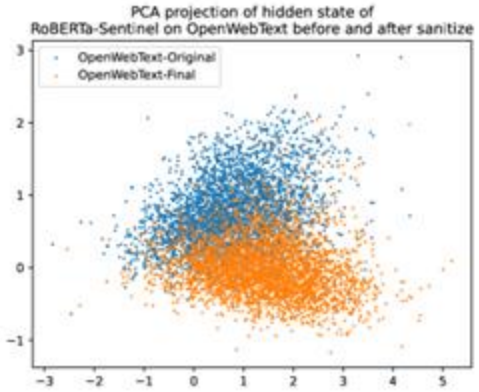
Fine-tuned LM achieved = 90% accuracy

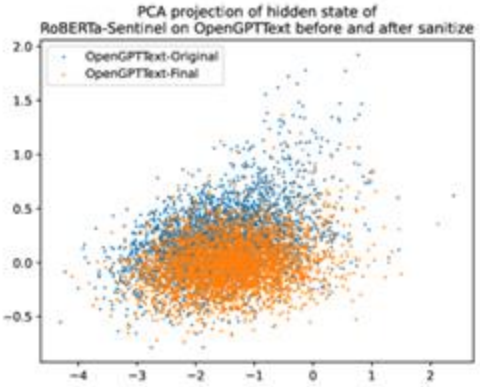
Dataset Collection: OpenGPTText

Subset	OpenGPTText	OpenWebText	Failed to Rephrase	Percentage
urlsf_00	3,888	391,590	27	0.99%
urlsf_01	3,923	392,347	0	1.00%
urlsf_02	3,260	391,274	652	0.83%
urlsf_03	3,891	390,161	10	1.00%
urlsf_04	3,684	390,250	218	0.94%
urlsf_05	3,602	389,874	296	0.92%
urlsf_06	3,494	390,339	409	0.90%
urlsf_09	3,653	389,634	243	0.94%
Total	29,395	3,125,469	1,885	0.94%

- No public dataset systematically containing ChatGPT outputs existed
- The author built OpenGPTText, a pair of human-written and ChatGPT-paraphrased samples
- Source: OpenWebText- web pages shared on Reddit, originally all human-written
- Process: Each paragraph rephrase through ChatGPT-3.5turbo with prompt: Rephrase the following paragraph by paragraph
- Final dataset: 29,395 text pairs

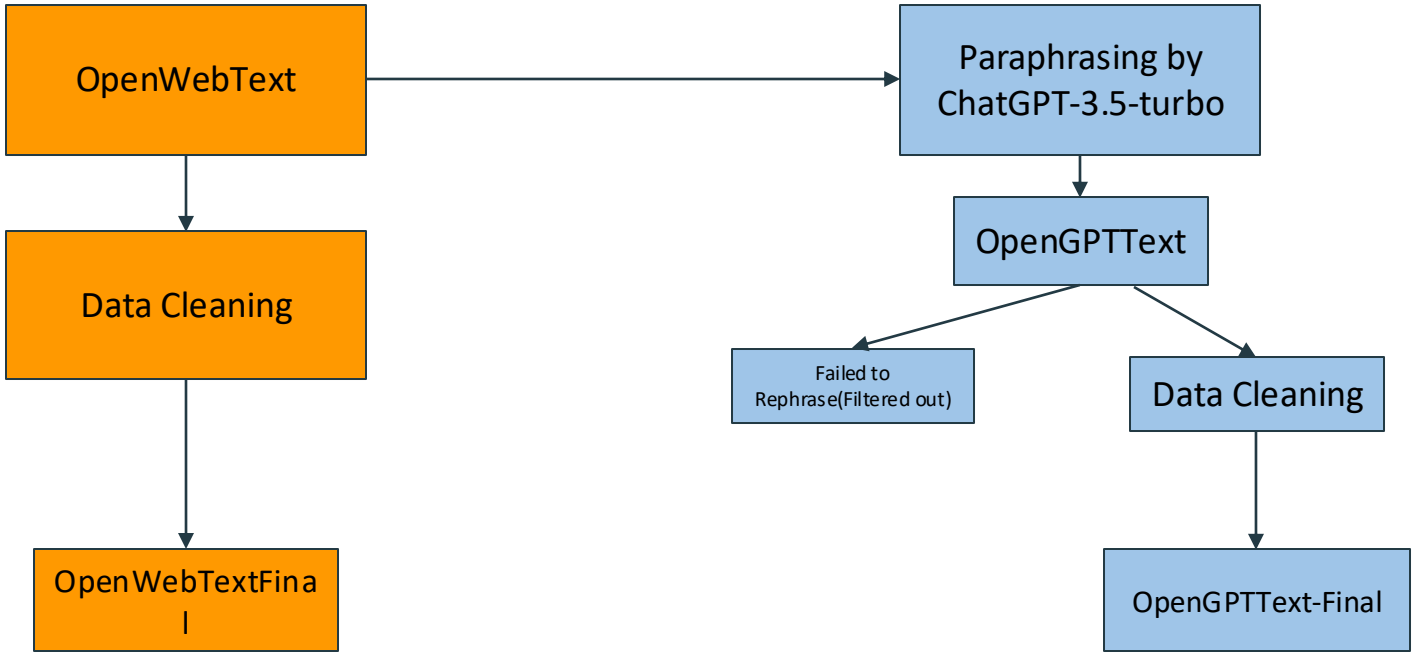
Data Cleaning



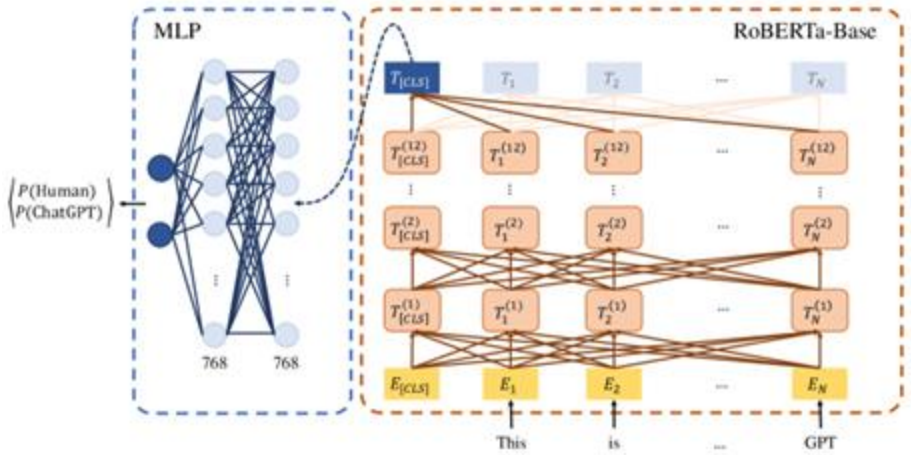


- ChatGPT outputs often contained stylistic artifacts
  - Smart quotes(“”) instead of ASCII(“”)
  - Inconsistent newlines(2-6 line breaks) and extra spaces
- Normalize both corpora with identical preprocessing
  - OpenWebText-Final (Human)
  - OpenGPTText-Final (ChatGPT)
- PCA indicates tighter clustering and clearer separation after cleaning

Dataset Roadmap

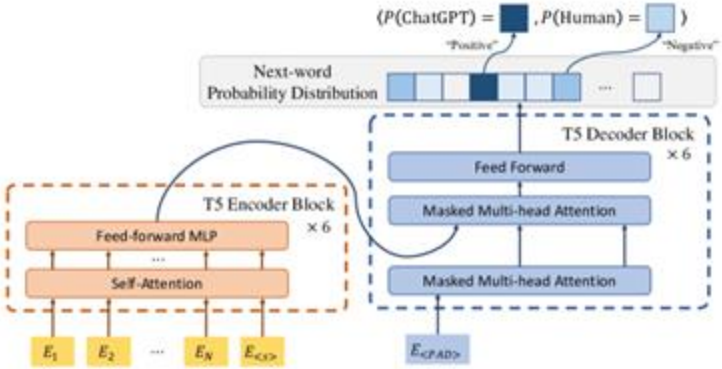


RoBERTa-Sentinel



1. Token Embeddings(E1~En):
   a. Each input word (Ex: This is GPT) is converted into numerical vectors
2. Frozen RoBERTa Encoder(12 Layers):
   a. Pre-trained Transformer layers process the tokens to capture contextual meaning. These layers are frozen, so their parameters are not updated during training.
3. [CLS] Token Representation:
   a. The special [CLS] token summarizes the meaning of the entire sentence
4. Feature Flow(Dashed Arrow):
   a. Information flows from RoBERTa to the MLP, but no gradient flows back
5. MLP Classifier (Trainable):
   a. The [CLS] embedding is fed into a 2 layer MLP that predicts: P(Human) or P(ChatGPT)

T5-Sentinel



- Token Embeddings(E1~En):
  - Each input word is tokenized and embeded
- T5 Encoder(X6 blocks, Trainable):
  - Self-attention + feed forward layers build contextual representations of whole sentence
  - Outputs a sequence of hidden states used by decoder
- Decoder Start & Masked Self-Attention
  - The decoder is primed with a <PAD>(start) token
  - Masked self attention lets the decoder look only at already generated positions
- Cross Attention to Encoder Outputs
  - The decoder attends to encoder hidden states to condition generation on the input text
  - This ties the prediction to the full sentence meaning
- Predict the Label Word
  - Positive -> ChatGPT generated
  - Negative-> Human written

Training and Evaluation

Hyper-Parameters	RoBERTa-Sentinel	T5-Sentinel
Epoch	15	5
Batch Size	512	512
Learning Rate	1×10^{-4}	5×10^{-4}
Weight Decay	1×10^{-3}	1×10^{-3}
Optimizer	AdamW	AdamW
Loss Function	Cross entropy	Cross entropy
Scheduler	Cosine annealing	Cosine annealing
Data Set	OpenGPTText-Final	OpenGPTText-Final

- Both models were trained on the OpenGPTText-Final dataset but with different training strategies and parameter scales which indicates how model size, learning rate and training scope affect performance and stability.
- RoBERTa-Sentinel trained much longer than T5 because only the MLP classifier was learning
- Higher learning rate in T5 helps the large model learn quickly without underfitting, a smaller learning rate RoBERTa prevents instability in smaller classifier head

Results(F1 Score, False Positive Rate and False Negative Rate)

Model	OpenGPTText-Final			OpenGPTText			GPT2-Output		
	F1	FPR	FNR	F1	FPR	FNR	F1	FPR	FNR
T5	**0.98**	**2.8**	**1.3**	**0.98**	3.5	**1.3**	0.06	**5.9**	96.7
RoBERTa	0.94	9.0	3.2	0.89	21.6	1.9	0.16	17.2	89.6
ZeroGPT	0.43	26.3	65.0	0.40	16.5	71.3	0.14	23.4	90.5
OpenAI-Detector	0.32	4.9	79.8	0.26	**1.6**	85.2	0.66	13.6	44.0
GPT2	0.23	2.8	86.8	0.22	4.1	87.2	**0.93**	6.4	**7.4**

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

$$\text{TPR} = \frac{TP}{TP + FN} \qquad \text{FPR} = \frac{FP}{FP + TN}$$

$$\text{TNR} = \frac{TN}{TN + FP} \qquad \text{FNR} = \frac{FN}{FN + TP}$$

- Both T5-Sentinel and RoBERTa-Sentinel achieve excellent accuracy on the OpenGPTText datasets (before and after cleaning) but sho w poor performance on GPT2-Output due to its higher randomness and stylistic differences from ChatGPT text.
- Additionally, GPT2-Detector fails to generalize from GPT2-based detection to ChatGPT detection, showing limited cross-model transfer ability.

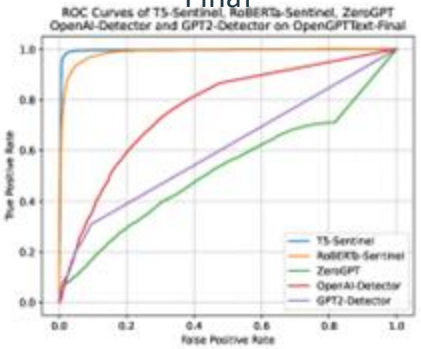
Results(AUC Value for each combination of data set and model)

Model	OpenGPTText-Final	OpenGPTText	GPT2-Output
T5-Sentinel	**0.993**	**0.992**	0.463
RoBERTa-Sentinel	0.986	0.976	0.423
ZeroGPT	0.526	0.555	0.413
OpenAI-Detector	0.765	0.752	0.770
GPT2-Detector	0.610	0.600	**0.976**

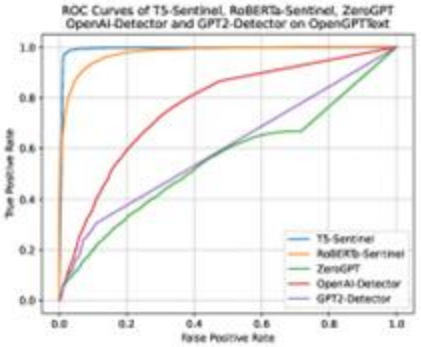
- T5-Sentinel and RoBERTa-Sentinel achieve extremely high AUC scores (above 0.97) on the OpenGPTText datasets, demonstrating strong detection performance for ChatGPT-generated text.
- Both models perform poorly on the GPT2-Output dataset, indicating limited generalization, whereas GPT2-Detector performs well only on GPT2 outputs but fails to detect ChatGPT text effectively.

Results(ROC Curve)

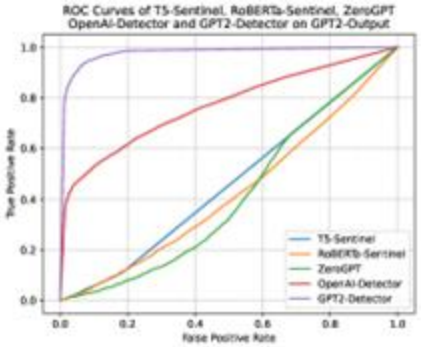
OpenGPT Text Final



OpenGPT Text



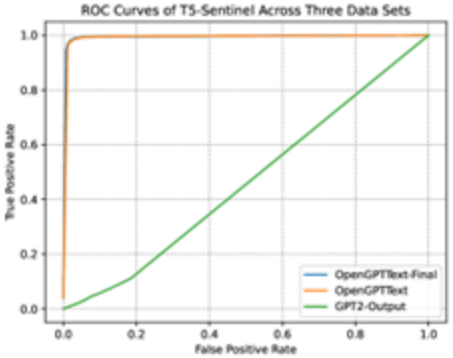
GPT2-Output



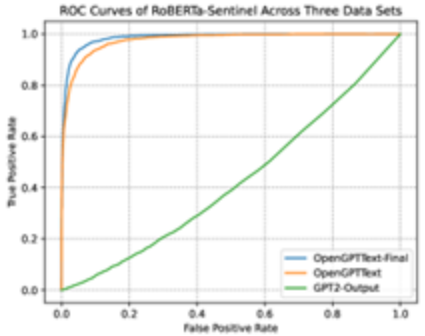
- The blue curve, T5 Sential dominates all others with the highest recall and lowest error.
- RoBERTa follows closely behind, while public detectors like ZeroGPT and OpenAI's detector lag far below
- On GPT-2 output, all models drop in accuracy
- GPT-Sentinel is excellent in domain but not yet generalize well to unseen LLM

Results(ROC Curve)

T5-Sentinel



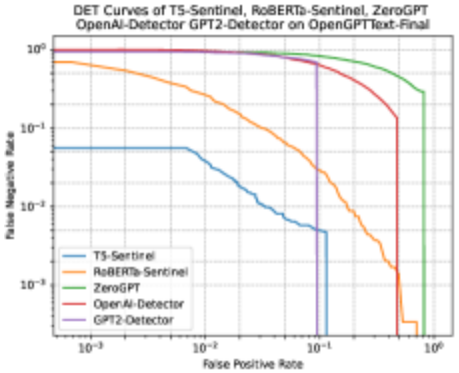
RoBERTa-Sentinel



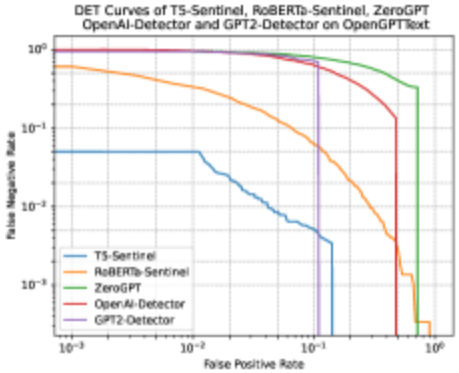
- T5-Sentinel and RoBERTa-Sentinel both achieve excellent performance on the OpenGPTText and OpenGPTText-Final datasets their ROC curves are nearly touching the top-left corner, showing high true positive rate (TPR) and low false positive rate (FPR).
- On the GPT2-Output dataset, however, both models perform poorly, with curves close to the diagonal line meaning their ability to detect GPT-2 generated text is weak.
- This demonstrates that ChatGPT-generated (paraphrased) text has distinct statistical and linguistic characteristics compared to GPT-2 text, leading to strong dataset specificity but limited generalization.

Results(DET Curve)

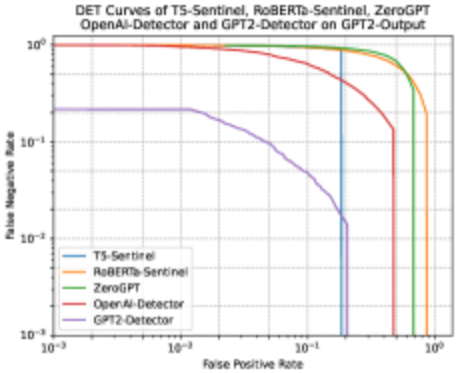
OpenGPT Text



OpenGPT Text



GPT2-Output

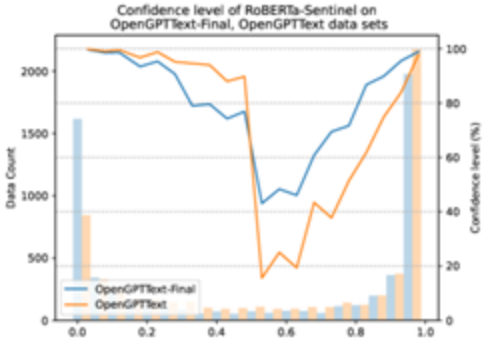


- Starting from Left both Sentinel model perform strongly with very low errors, but there are more variation due to noise and artifacts before cleaning and for GPT2 it confirms limited cross model generalization
- GPT-Sentinel excels in domain but remains model specific

Confidence

T5-Sentinel

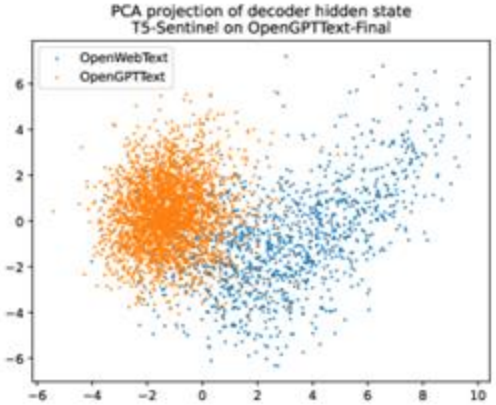
RoBERTa-Sentinel



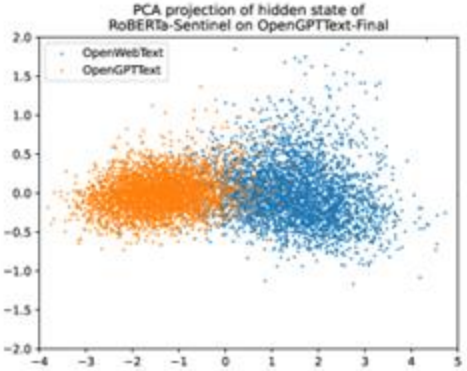
- T5 Sentinel shows sharp confidence peaks near 0 and 1 which is very decisive predictions
- Data cleaning increases confidence stability and reduce the ambiguity
- Overall, it confirms that cleaning and fine-tuning lead to a more trustworthy detection model

Interpretability(PCA)

T5-Sentinel

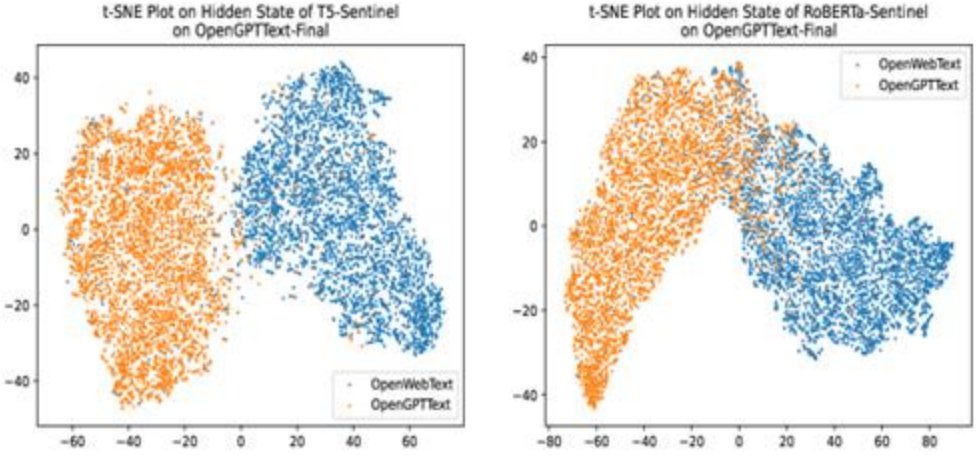


RoBERTa-Sentinel



- To better understand how the models differentiate human vs ChatGPT-generated text, a PCA visualization was performed on the hidden representations of both models.
- RoBERTa-Sentinel: Hidden states were extracted from the last layer of the MLP.
- T5-Sentinel: Hidden states were taken from the final decoder block output.
- The PCA used test samples from the OpenGPTText-Final dataset.
- Both models formed two distinct clusters one for human-written text and one for ChatGPT-rephrased text.
- This clustering shows that the models learned implicit linguistic patterns that separate human and AI writing styles.
- T5-Sentinel's clusters are more distinct, indicating stronger discriminative power than RoBERTa-Sentinel.

Interpretability



- t-SNE visualize each text hidden representation in 2D
- T5-Sentinel forms two clear non overlapping cluster
- RoBERT-Sentinel shows partial overlap which shows weaker separation
- It confirms that T5 learns deeper semantic distinctions, not just token patterns

Future work & Conclusion

Future work

- Current models are only trained in English so develop with different languages
- Plan to collect the diverse datasets to evaluate model accuracy across different textual contexts and task types

Conclusion

- Introduced OpenGPTText, a high quality dataset rephrased by ChatGPT for AI vs human
- Achieved over 97% accuracy on test using F1, AUC, and ROC metrics
- Conducted interpretability analysis(PCA & tSNE) to revealed clear separation between human and AI text
- Highlighted the potential for real world AI content detection

Thank You!