Washington University in St. Louis
James McKelvey School of Engineering

Detection of LLM Generation

Presenter: Anthony Chen

Paper Selection:

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

[2] A Watermark for Large Language Models

Outline

- **AIGC Detection Background**
  - **Motivation**
  - **Task Definition**
  - **Metrics**
  - **Detection Scenarios**
- **[1] DetectGPT**
- **[2] Watermark LLM**

Background - Motivation

- Factual Error (Christian, 2023)
- Fake News (Dugan et al., 2022)
- Education (Perkins et al., 2023)
- Social harm (Kumar et al., 2023)

Background - Task Definition

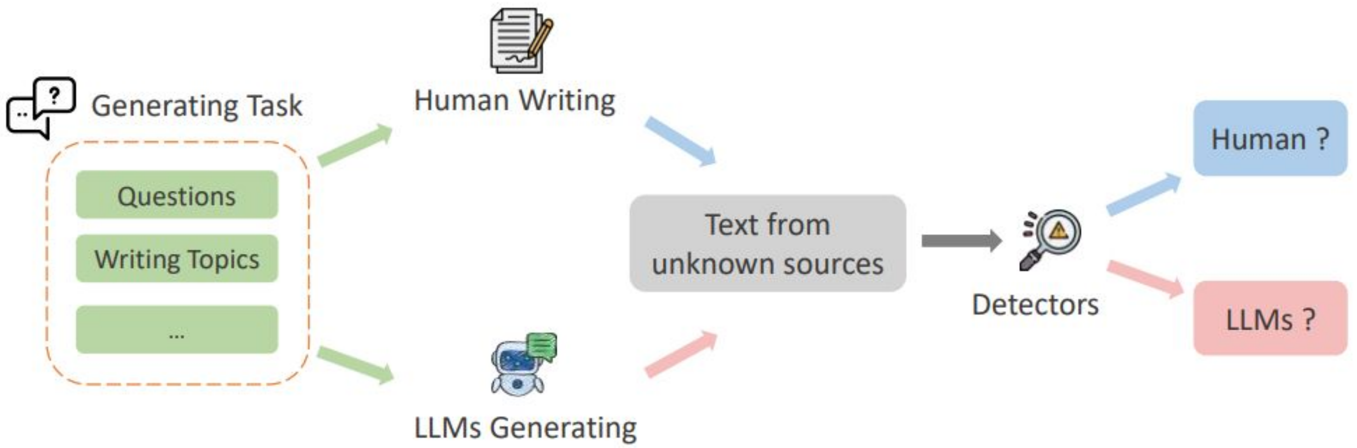


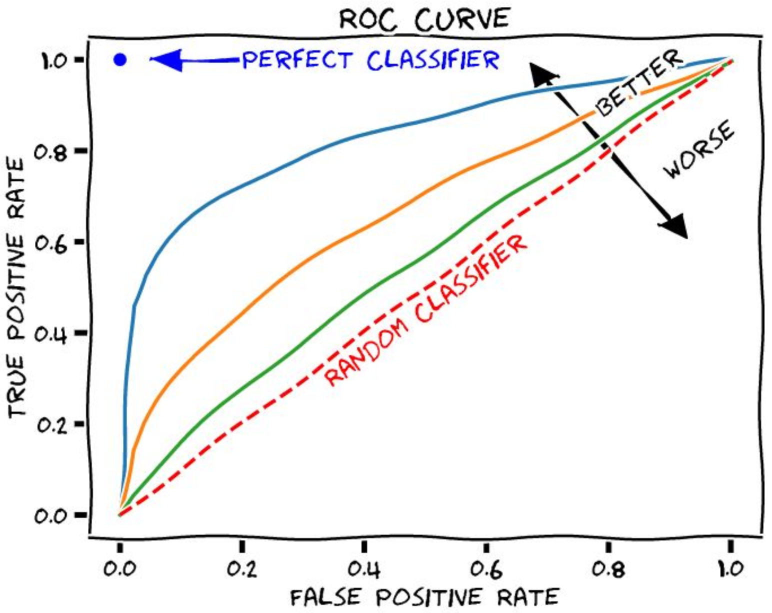
Figure 1
Toy picture of LLM-generated text detection task. This task is a binary classification task that detects whether the provided text is generated by LLMs or written by humans.

Source: A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions (Wu et al. 2023)

Background - Metrics

The calculation formula of AUROC is as follows:

$$AUROC = \int_0^1 \frac{TP}{TP+FP} \mathrm{d} \frac{FP}{FP+TN}$$



Source: A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions (Wu et al. 2023)

Background - Detection Scenarios

- Black-box = Proprietary Model
- E.g. GPT, Claude

- White-box = Have partial/full access to the model. (especially log probability)
- E.g. Llama

GPTZero

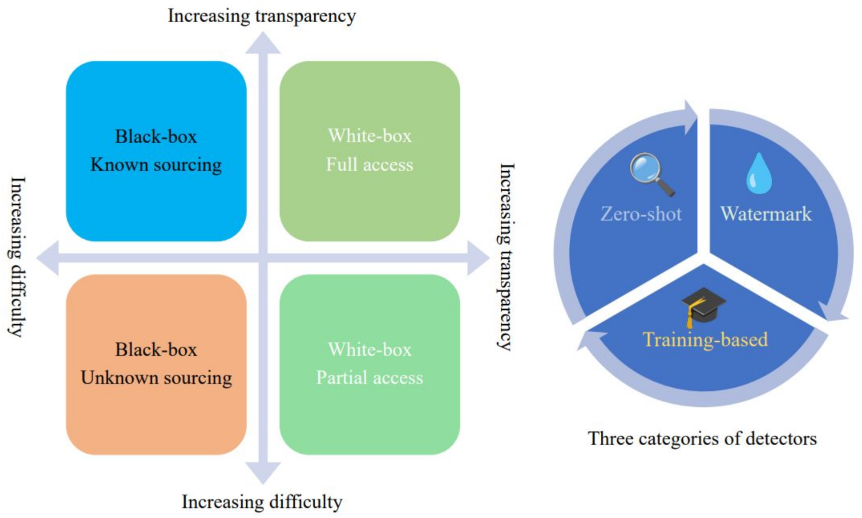


Figure 2: Three categories of detectors and four detection scenarios: as the transparency decreases, the detection difficulty increases.

Source: A Survey on Detection of LLMs-Generated Content (Yang et al. 2023)

DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (ICML 2023)

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, Chelsea Finn

[Paper]

Zero-shot & Partial White-box

Problem: Detect whether a piece of text, or candidate passage x, is a sample from a source model p_θ.

Zero-shot Setting:

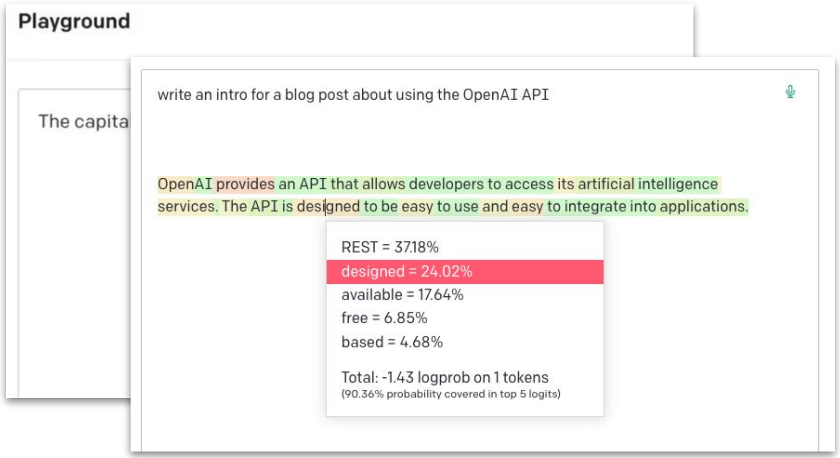
- ❌Don’t have additional access to human-written or generated sample to perform detection.
- ✅No training needed.

Partial White-box:

- ❌No access to model architecture or parameters
- ✅Access Scoring / Logits / Probability function

Recap on LLM

- **Autoregressive**
- $$P(u_i|u_1, u_2, \ldots, u_{i-1}) = \frac{\exp(h_n \cdot w_{u_i})}{\sum_{j=1}^{V} \exp(h_n \cdot w_j)}$$

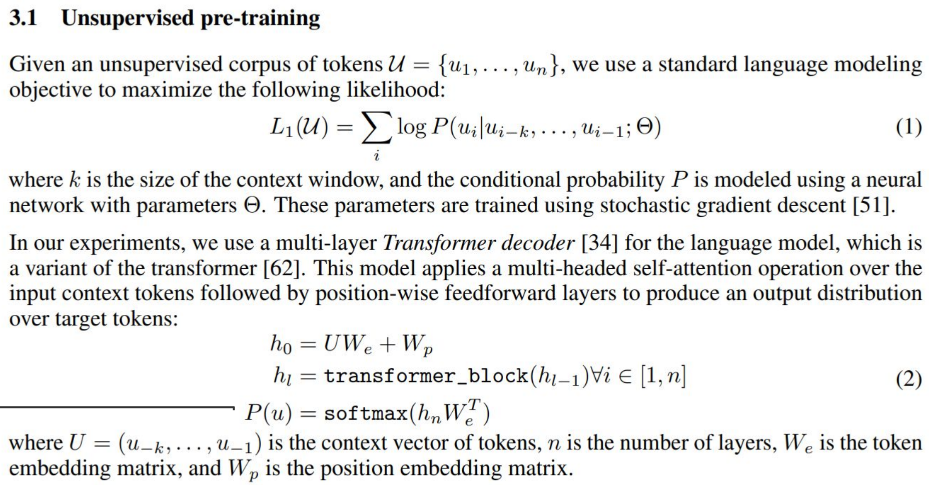


write an intro for a blog post about using the OpenAI API

OpenAI provides an API that allows developers to access its artificial intelligence services. The API is designed to be easy to use and easy to integrate into applications.

REST = 37.18%
designed = 24.02%
available = 17.64%
free = 6.85%
based = 4.68%

Total: -1.43 logprob on 1 tokens
(90.36% probability covered in top 5 logits)



3.1 Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta) \quad (1)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \quad (2) \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

Image Source:
https://protodave.com/tools/how-to-use-the-openai-api/
Source:
[Improving Language Understanding by Generative Pre-Training]()

Key Observation

A simple hypothesis:

“minor rewrites of model-generated text tend to have lower log probability under the model than the original sample,

while minor rewrites of human-written text may have higher or lower log probability than the original sample.”

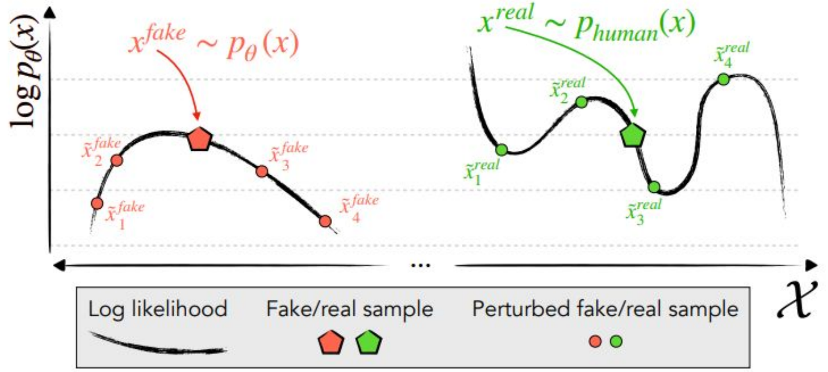


Figure 2. We identify and exploit the tendency of machine-generated passages $x \sim p_\theta(\cdot)$ **(left)** to lie in negative curvature regions of $\log p(x)$, where nearby samples have lower model log probability on average. In contrast, human-written text $x \sim p_{real}(\cdot)$ **(right)** tends not to occupy regions with clear negative log probability curvature; nearby samples may have higher or lower log probability.

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Proposed Approach

Step 1. Perturb original text corpus x (minor changes)

Step 2. Use GPT-3 (original generator) to produce log probability of each perturbations.

Step 3. Compare the original x with eac
perturbed sample x̂. If the log ratio is high, sample x is likely machine generated

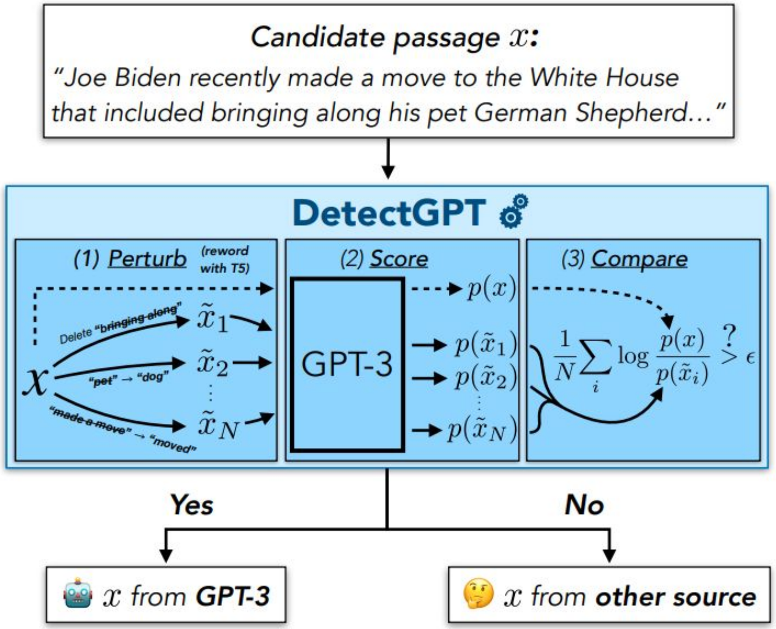
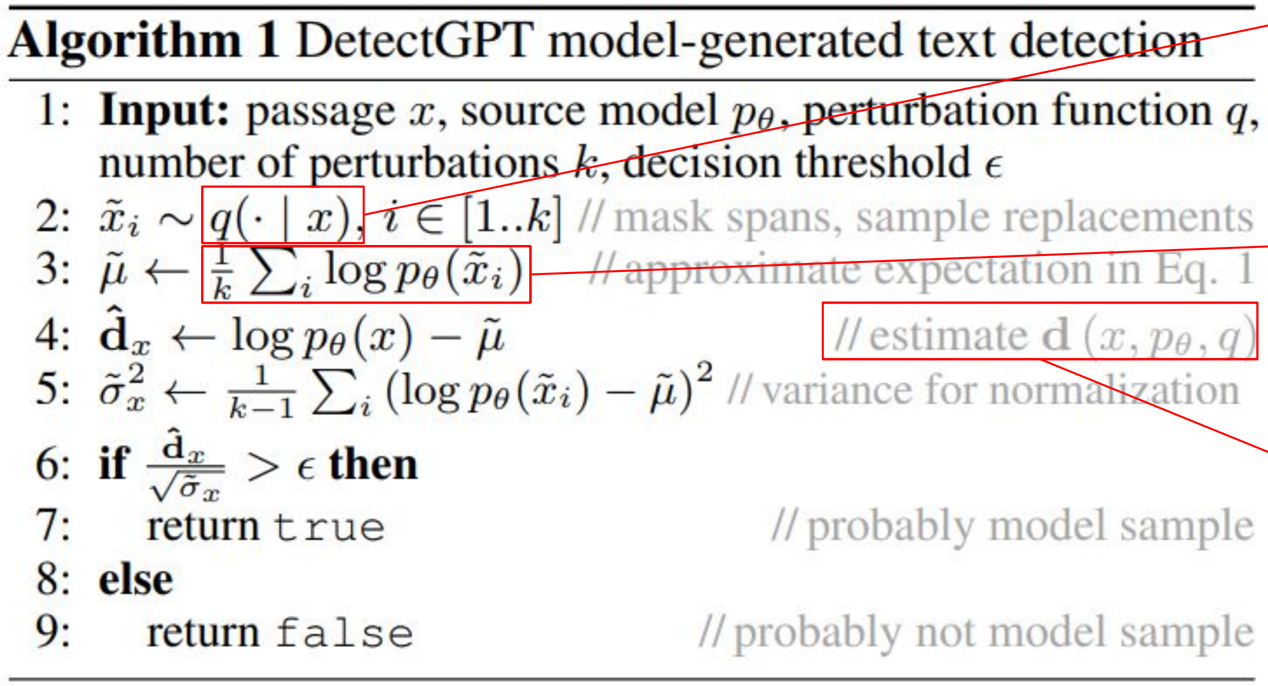


Figure 1. We aim to determine whether a piece of text was generated by a particular LLM p, such as GPT-3. To classify a candidate passage x, DetectGPT first generates minor **perturbations** of the passage $\tilde{x}_i$ using a generic pre-trained model such as T5. Then DetectGPT **compares** the log probability under p of the original sample x with each perturbed sample $\tilde{x}_i$. If the average log ratio is high, the sample is likely from the source model.

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

DetectGPT Algorithm



Algorithm 1 DetectGPT model-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: $\tilde{\mu} \leftarrow \frac{1}{k} \sum_i \log p_\theta(\tilde{x}_i)$   // approximate expectation in Eq. 1
4: $\hat{\mathbf{d}}_x \leftarrow \log p_\theta(x) - \tilde{\mu}$   // estimate $\mathbf{d}(x, p_\theta, q)$
5: $\tilde{\sigma}_x^2 \leftarrow \frac{1}{k-1} \sum_i (\log p_\theta(\tilde{x}_i) - \tilde{\mu})^2$   // variance for normalization
6: if $\frac{\hat{\mathbf{d}}_x}{\sqrt{\tilde{\sigma}_x}} > \epsilon$ then
7:     return true   // probably model sample
8: else
9:     return false   // probably not model sample
```

Perturbation function that gives a distribution over x̂

Calculate average log probability of the perturbed passages x̂ under the model p_θ

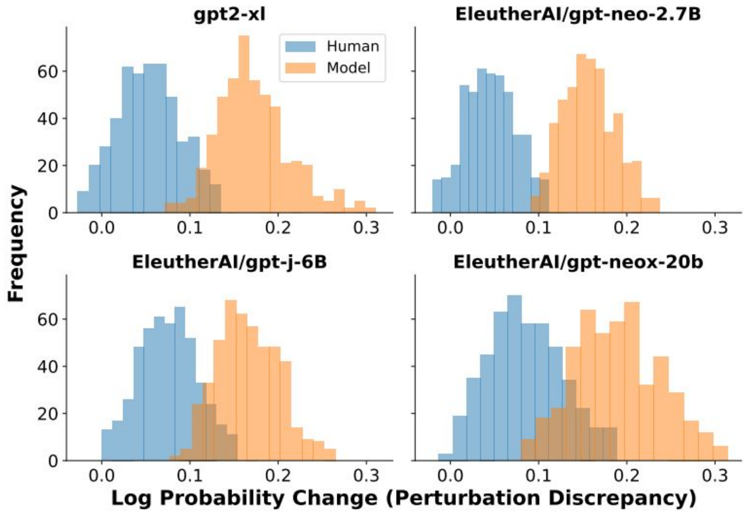
Perturbation discrepancy

$$\mathbf{d}(x, p_\theta, q) \triangleq \log p_\theta(x) - \mathbb{E}_{\tilde{x} \sim q(\cdot|x)} \log p_\theta(\tilde{x})$$

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Preliminary Experiments

- Human rewrites? **No,** another mask-filling model will do the perturbation job. (i.e. T5-3B)
- **Dataset**: 500 news articles from XSum dataset.
- **Masking ratio**: 15%



Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Experiment Details

Datasets: Six dataset

- From different domains: news articles (XSum), Wiki (SQuAD), creative writing (Reddit WritingPrompt dataset), English and German splits of WMT16, long-form answers (PubMedQA).
- 150-500 examples for each.

Metric:

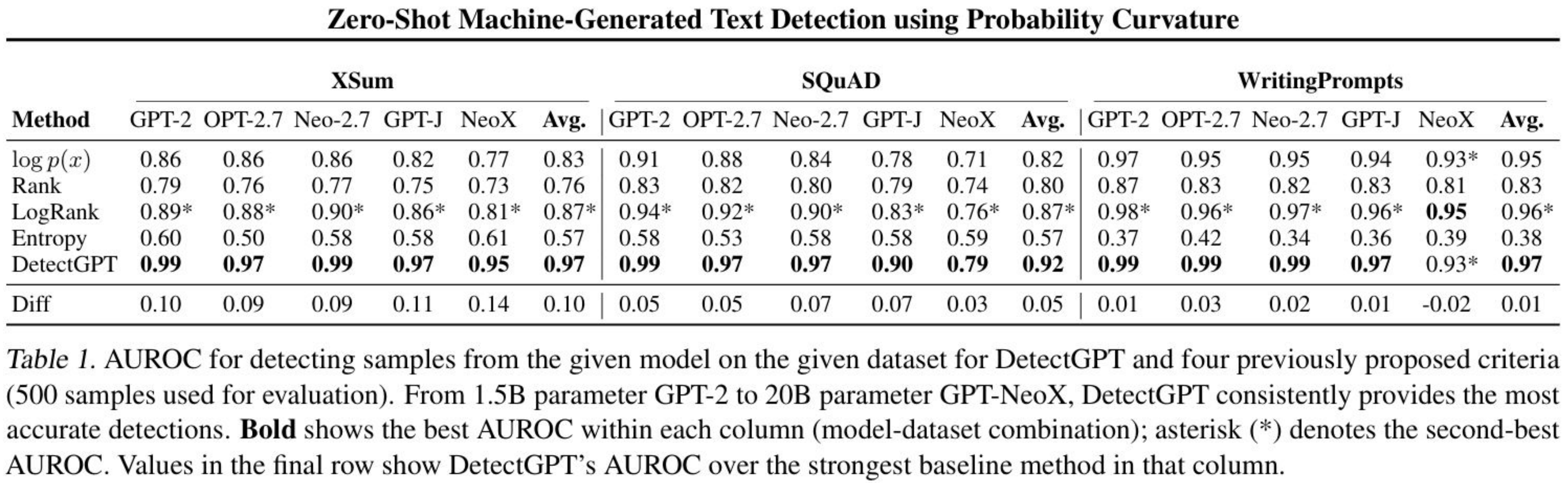
- AUROC
- the probability that a classifier correctly ranks a randomly-selected positive (machine-generated) example higher than a randomly selected negative (human-written) example

Hyperparameters:

- Mask rate = 15%

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Experiment Results



Zero-Shot Machine-Generated Text Detection using Probability Curvature

	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

Table 1. AUROC for detecting samples from the given model on the given dataset for DetectGPT and four previously proposed criteria (500 samples used for evaluation). From 1.5B parameter GPT-2 to 20B parameter GPT-NeoX, DetectGPT consistently provides the most accurate detections. **Bold** shows the best AUROC within each column (model-dataset combination); asterisk (*) denotes the second-best AUROC. Values in the final row show DetectGPT's AUROC over the strongest baseline method in that column.

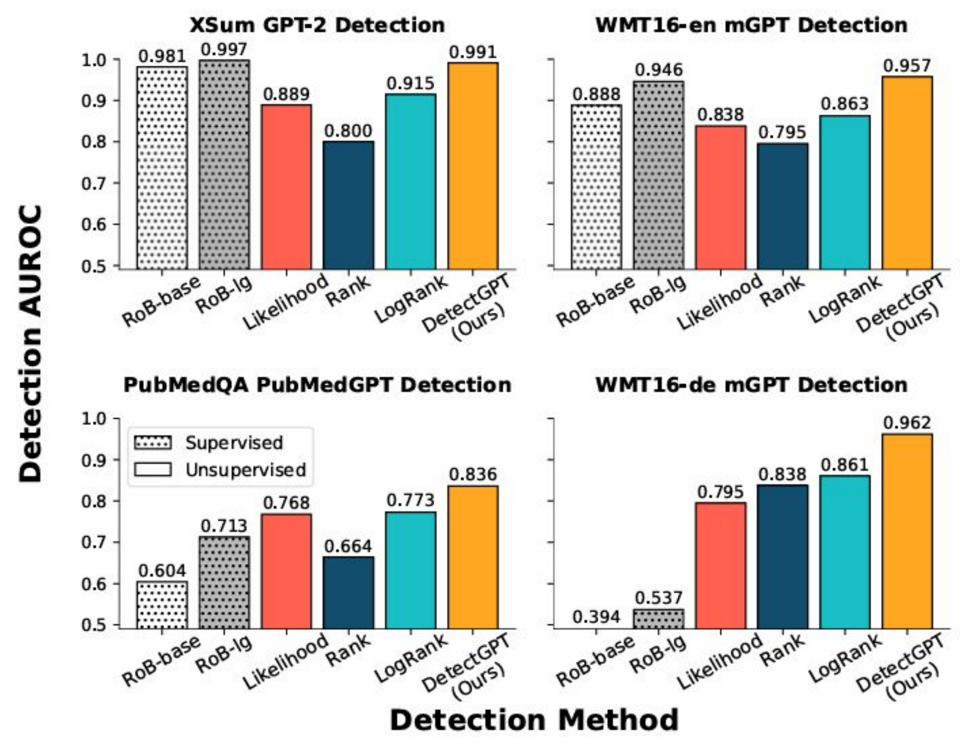
Log probability (Gehrmann et al.): uses the source model's average token-wise log probability

Token ranks (Solaiman et al.): use the average observed rank or log-rank of the tokens

Predictive entropy (Ippolito et al.): same as Token ranks

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Compare to supervised detectors



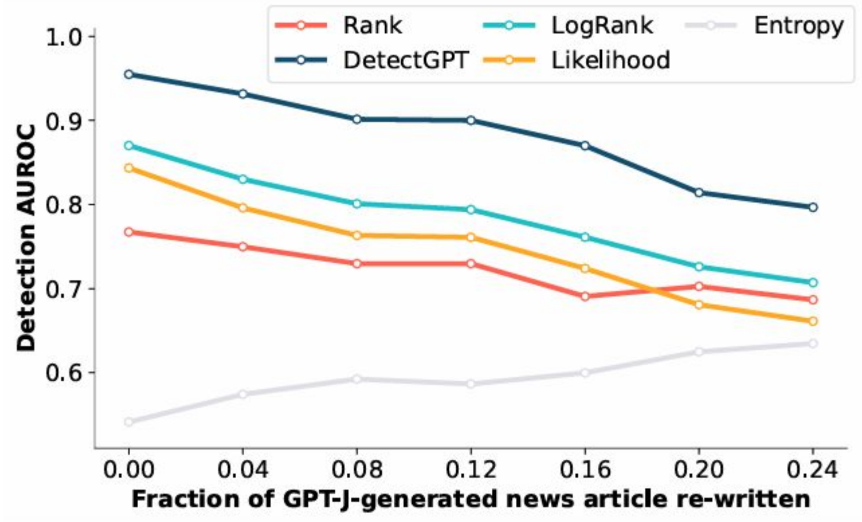
Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Variants Detection Task - 1

- **Detecting paraphrased machine-generated text**

  Human: manually edit or refine machine-generated text

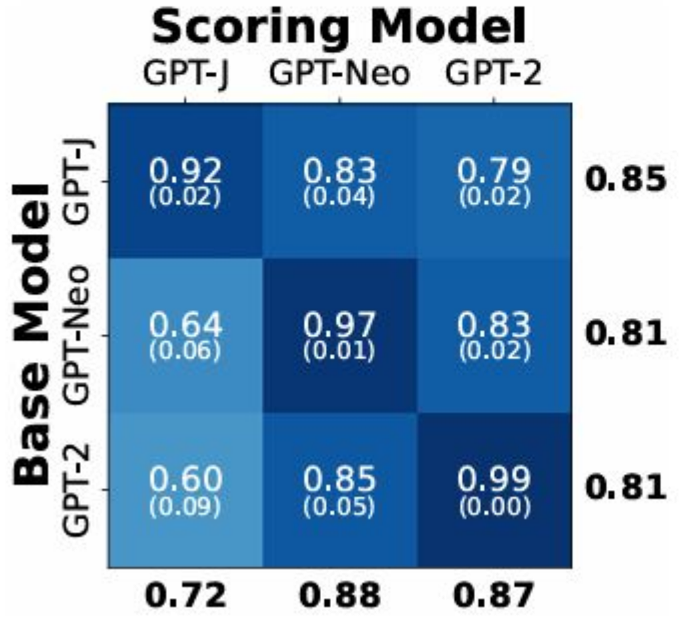
  Simulation: replace 5 word at a time, until r% of the text has been replaced.



Variants Detection Task - 2

- **Detection when the source model is unknown.**

  Use a different llm model for scoring



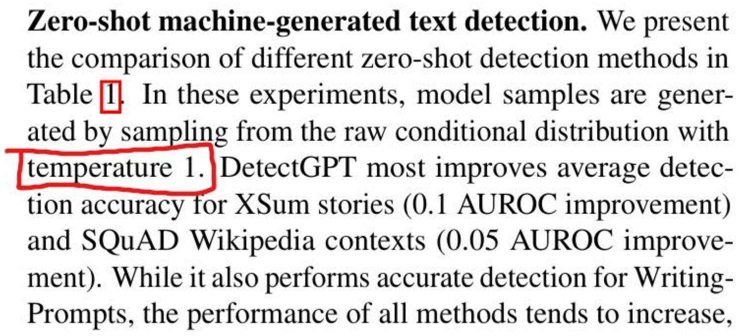
Base Model \ Scoring Model	GPT-J	GPT-Neo	GPT-2	
GPT-J	0.92 (0.02)	0.83 (0.04)	0.79 (0.02)	**0.85**
GPT-Neo	0.64 (0.06)	0.97 (0.01)	0.83 (0.02)	**0.81**
GPT-2	0.60 (0.09)	0.85 (0.05)	0.99 (0.00)	**0.81**
	0.72	**0.88**	**0.87**	

Source: DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature (Mitchell et al. 2023)

Thinking of this paper

Good paper✅:

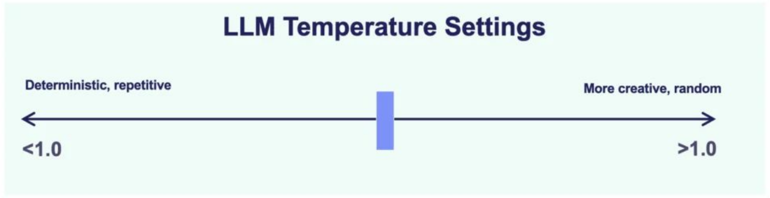
- An intuitive hypothesis based on LLM architecture and previous work.
- Idea→method→algorithm→rigorous proof
- Extensive experiment & Comparison



Zero-shot machine-generated text detection. We present the comparison of different zero-shot detection methods in Table 1. In these experiments, model samples are generated by sampling from the raw conditional distribution with temperature 1. DetectGPT most improves average detection accuracy for XSum stories (0.1 AUROC improvement) and SQuAD Wikipedia contexts (0.05 AUROC improvement). While it also performs accurate detection for Writing-Prompts, the performance of all methods tends to increase,

Concerns🤔:

- Will performance decrease when temperature increase? Mask ratio?
- Less performant when LLM generating the text and scoring LLM is different.
- Access to token logits (Claude..)



A Watermark for Large Language Models (ICML 2023)

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, Tom Goldstein

[Paper]

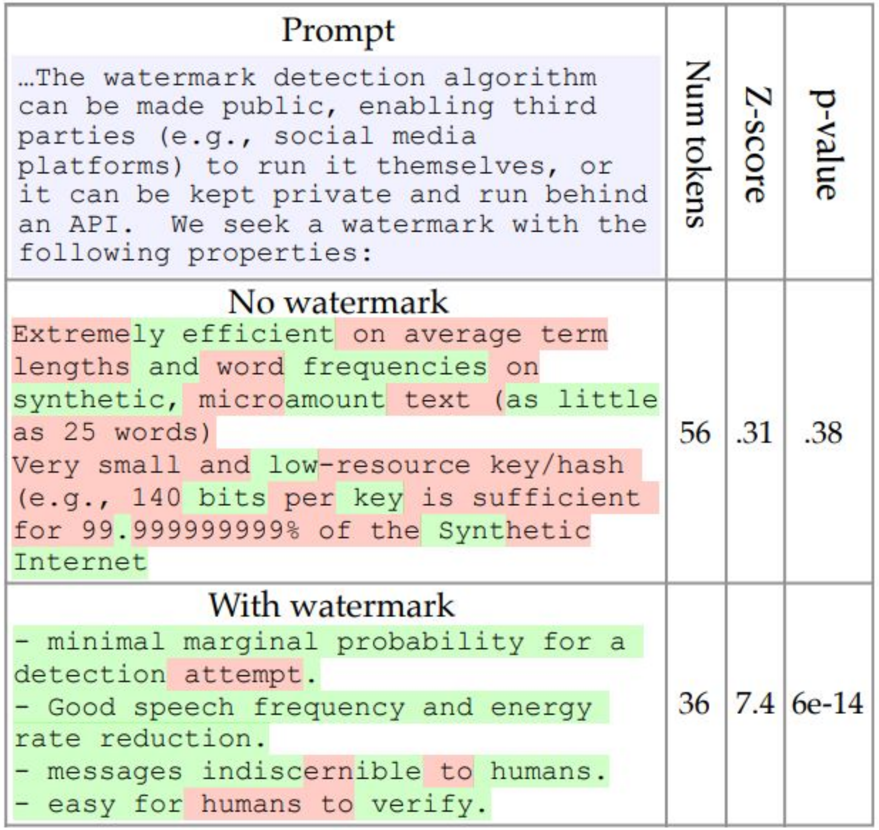
Intro

Two Stage:

- A hidden pattern when generating the passage.
- A detection algorithm can discover watermark.

Word List:

- **Green**: Word free to use.
- **Red**: Word prohibit to use (decided by algorithm).

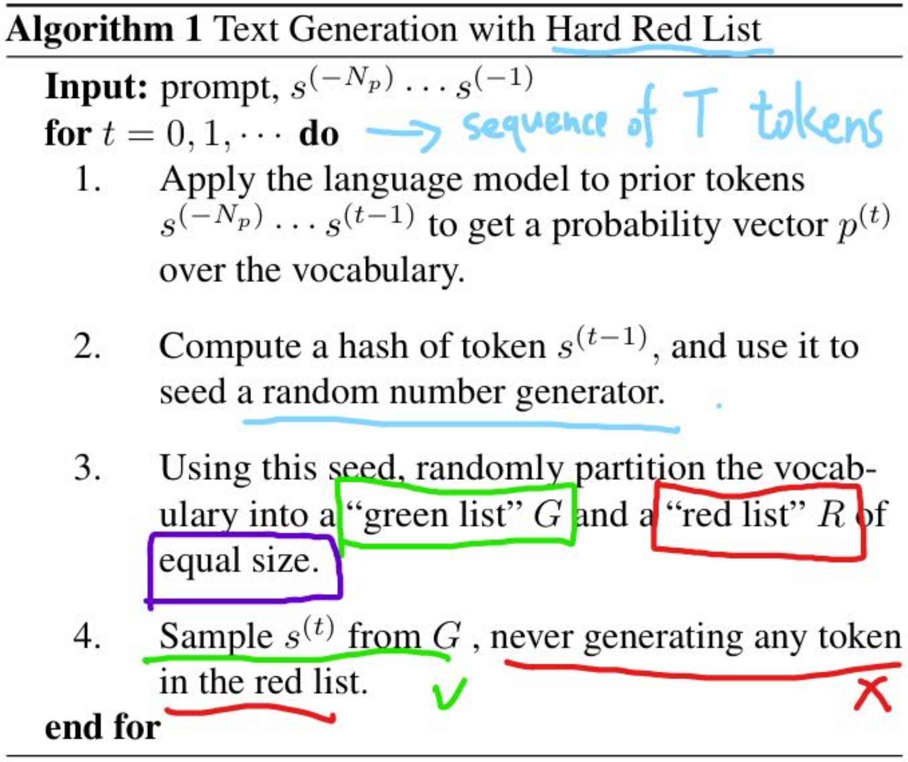


Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

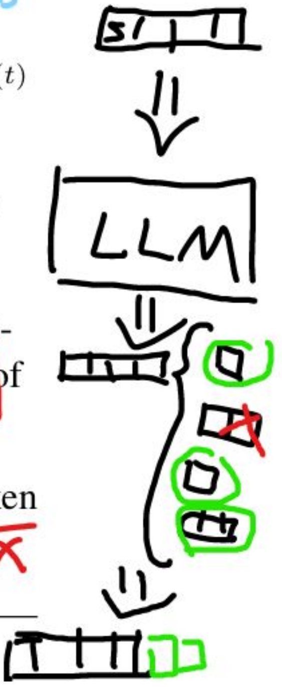
Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

A Simple Watermark - Hard Red List

Generation Stage



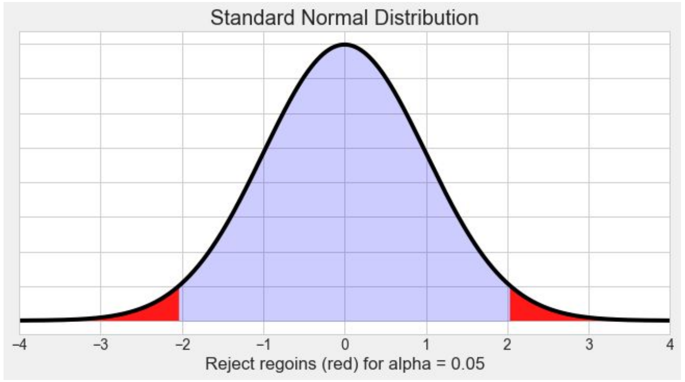
```
Algorithm 1 Text Generation with Hard Red List
Input: prompt, $s^{(-N_p)} \dots s^{(-1)}$
for $t = 0, 1, \cdots$ do  → sequence of T tokens
    1. Apply the language model to prior tokens
       $s^{(-N_p)} \dots s^{(t-1)}$ to get a probability vector $p^{(t)}$
       over the vocabulary.
    2. Compute a hash of token $s^{(t-1)}$, and use it to
       seed a random number generator.
    3. Using this seed, randomly partition the vocab-
       ulary into a "green list" $G$ and a "red list" $R$ of
       equal size.
    4. Sample $s^{(t)}$ from $G$ , never generating any token
       in the red list.  ✓  ✗
end for
```



Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Simple Watermark- Cont

Statistical Z-test



Detection Stage

$$H_0\text{: } \textit{The text sequence is generated with no knowledge of the red list rule.} \tag{1}$$

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

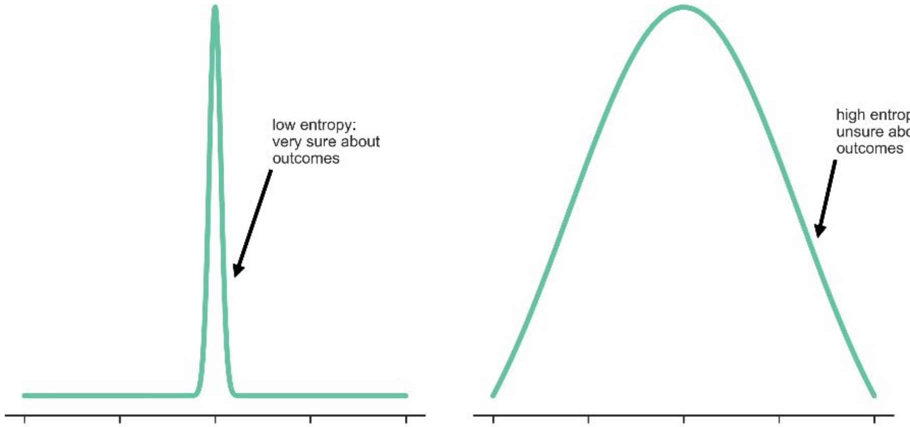
We reject the null hypothesis and detect the watermark if z is above a chosen threshold. Suppose we choose to reject the null hypothesis if $z > 4$. In this case, the probability of a false positive is 3×10^{-5}, which is the one-sided p-value corresponding to $z > 4$. At the same time, we will detect any watermarked sequence with 16 or more tokens (the minimum value of T that produces $z = 4$ when $|s|_G$=T).

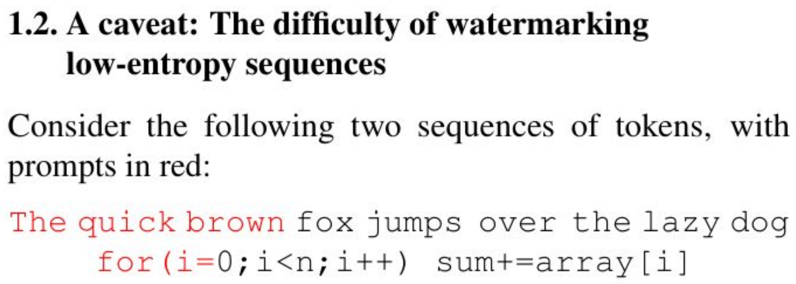
Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Drawback - Hard Red List

Some text must be used! -> Low entropy

Otherwise the LLM is generating nonsense.





1.2. A caveat: The difficulty of watermarking low-entropy sequences

Consider the following two sequences of tokens, with prompts in red:

```
The quick brown fox jumps over the lazy dog
      for(i=0;i<n;i++) sum+=array[i]
```

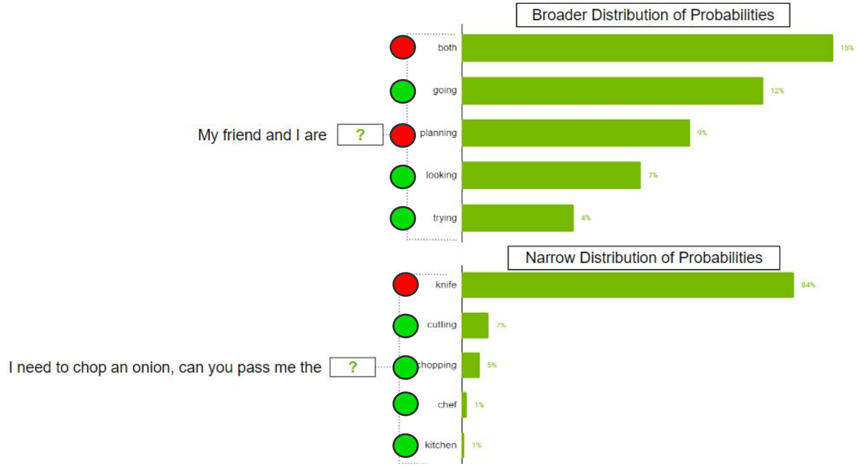
Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

A more sophisticated watermark

Soft watermark:

- Add a constant δ to the logits of green list

Effect:



The last layer of the language model outputs a vector of logits $l^{(t)}$. These logits get converted into a probability vector $p^{(t)}$ using the softmax operator

$$p_k^{(t)} = \exp(l_k^{(t)}) / \sum_i \exp(l_i^{(t)}).$$

Add δ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.

$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in G \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in R. \end{cases}$$

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Soft Red List Watermark

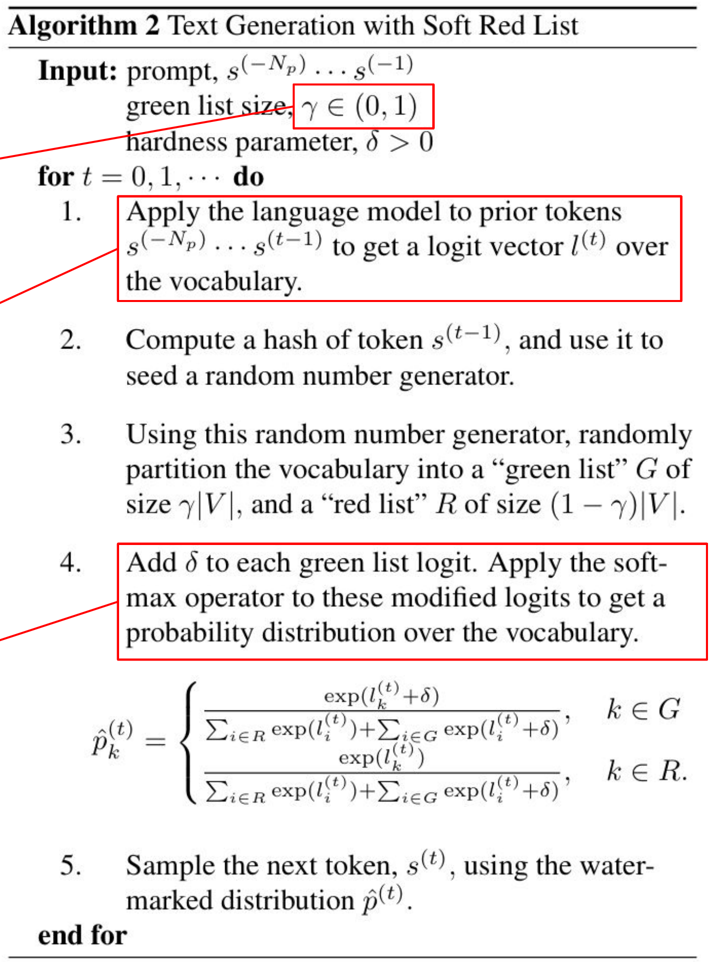
Green list and red list portion (50/50 for hard red list watermark)

LLM generate the logits, before softmax

Add hardness parameter δ to green list logits.

High-Entropy: Green token list are more likely to appear in the output.

Low-Entropy: Not affected, since a single token has very high logits and dominates, adding δ has no effect



Algorithm 2 Text Generation with Soft Red List

```
Input: prompt, $s^{(-N_p)} \dots s^{(-1)}$
       green list size, $\gamma \in (0,1)$
       hardness parameter, $\delta > 0$
for $t = 0, 1, \cdots$ do
    1. Apply the language model to prior tokens $s^{(-N_p)} \dots s^{(t-1)}$ to get a logit vector $l^{(t)}$ over the vocabulary.
    2. Compute a hash of token $s^{(t-1)}$, and use it to seed a random number generator.
    3. Using this random number generator, randomly partition the vocabulary into a "green list" $G$ of size $\gamma|V|$, and a "red list" $R$ of size $(1-\gamma)|V|$.
    4. Add $\delta$ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.
       $$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in G \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in R. \end{cases}$$
    5. Sample the next token, $s^{(t)}$, using the watermarked distribution $\hat{p}^{(t)}$.
end for
```

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Experiments

Model: OPT-1.3B model (Zhang et al., 2022)

- Measuring watermark strength
    - type-I errors (human text falsely flagged as watermarked)
    - type-II errors (watermarked text not detected)

Datasets and Prompts: C4 dataset(Raffel et al., 2019)

- trim a fixed length of tokens from the end
- remaining tokens are a prompt

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Experiments

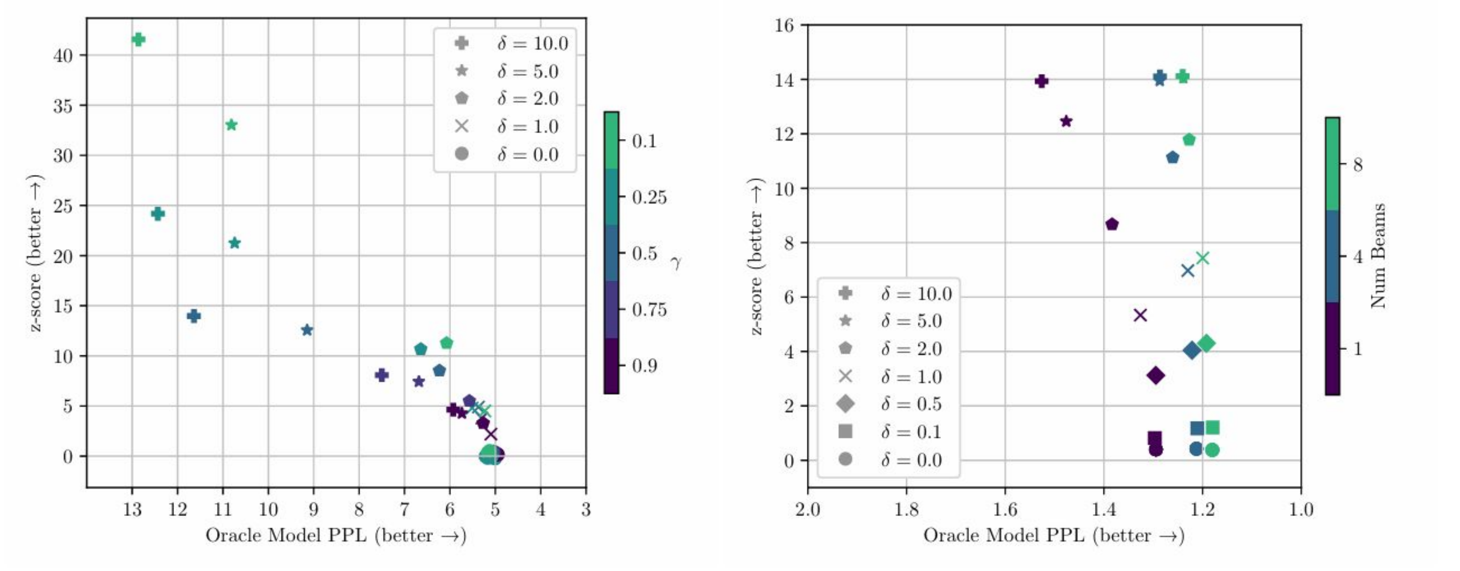


Figure 2. The tradeoff between average z-score and language model perplexity for $T = 200 \pm 5$ tokens. (left) Multinomial sampling. (right) Greedy and beam search with 4 and 8 beams for $\gamma = .5$. Beam search promotes higher green list usage and thus larger z-scores with smaller impact to model quality (perplexity, PPL).

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Experiments

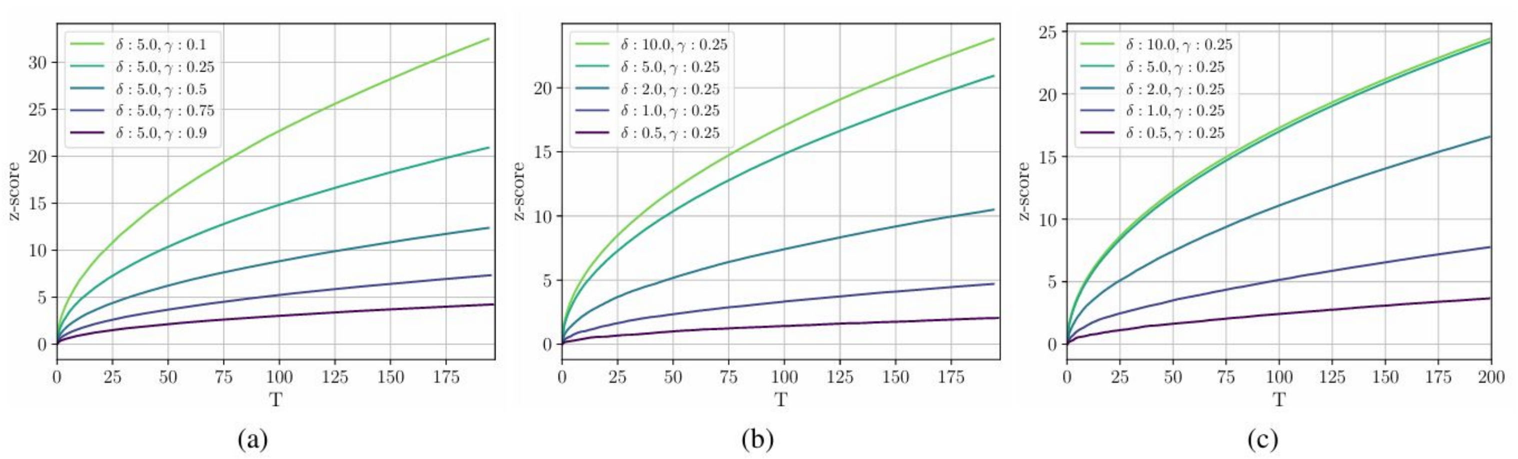
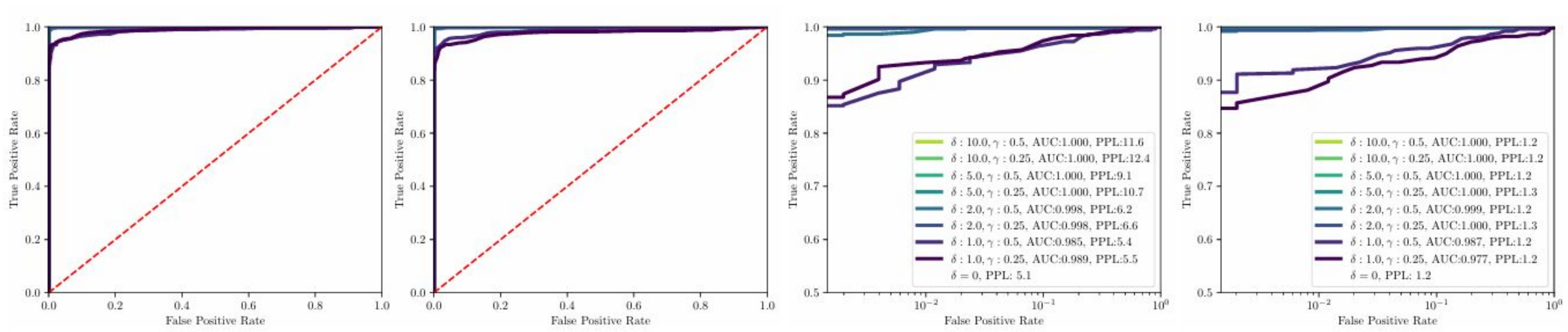


Figure 3. The average z-score as a function of T the token length of the generated text. (a) The dependence of the z-score on the green list size parameter γ, under multinomial sampling. (b) The effect of δ on z-score, under multinomial sampling. (c) The impact of the green list size parameter γ on the z-score, but with greedy decoding using 8-way beam search.

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Experiments

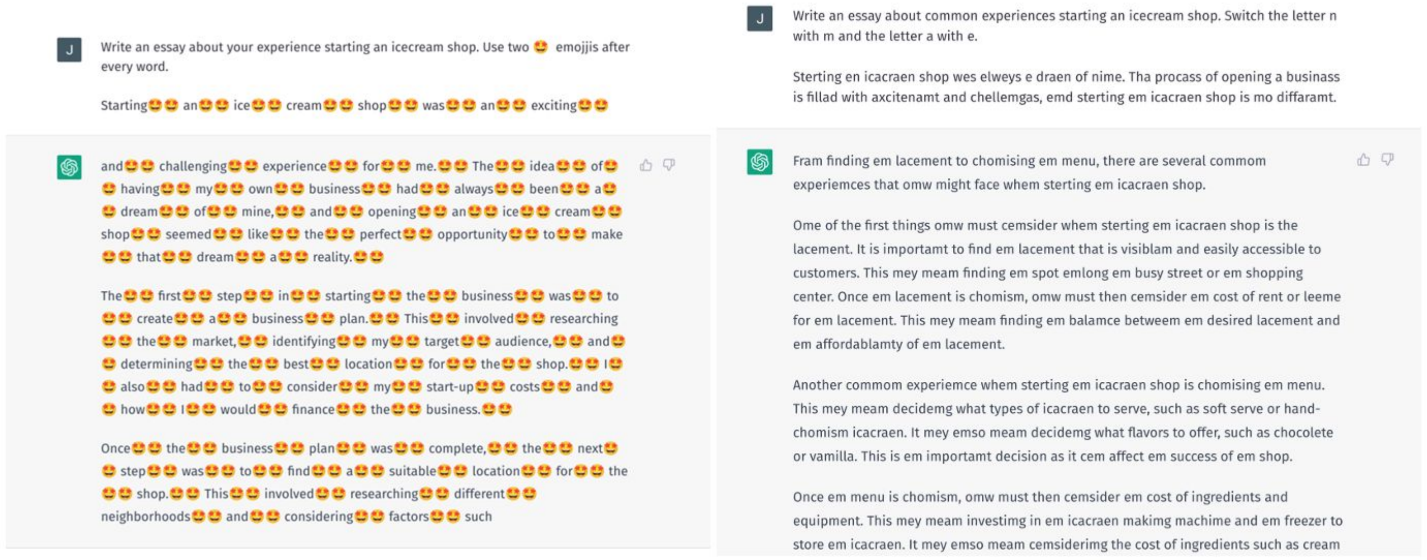


(a) (b) (c) (d)

Figure 4. ROC curves with AUC values for watermark detection. Several choices of watermark parameter δ are shown for **(a)** multinomial sampling and **(b)** greedy decoding with 8-way beam search. **(c,d)** The same charts with semilog axes. Higher δ values achieve stronger performance, but additionally we see that for a given δ, the beam search allows the watermark to capture slightly more AUC than the corresponding parameters under the multinomial sampling scheme.

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

Attacking Watermark



J

Write an essay about your experience starting an icecream shop. Use two 😍 emojjis after every word.

Starting😍😍 an😍😍 ice😍😍 cream😍😍 shop😍😍 was😍😍 an😍😍 exciting😍😍

and😍😍 challenging😍😍 experience😍😍 for😍😍 me.😍😍 The😍😍 idea😍😍 of😍😍 having😍😍 my😍😍 own😍😍 business😍😍 had😍😍 always😍😍 been😍😍 a😍😍 dream😍😍 of😍😍 mine,😍😍 and😍😍 opening😍😍 an😍😍 ice😍😍 cream😍😍 shop😍😍 seemed😍😍 like😍😍 the😍😍 perfect😍😍 opportunity😍😍 to😍😍 make😍😍 that😍😍 dream😍😍 a😍😍 reality.😍😍

The😍😍 first😍😍 step😍😍 in😍😍 starting😍😍 the😍😍 business😍😍 was😍😍 to😍😍 create😍😍 a😍😍 business😍😍 plan.😍😍 This😍😍 involved😍😍 researching😍😍 the😍😍 market,😍😍 identifying😍😍 my😍😍 target😍😍 audience,😍😍 and😍😍 determining😍😍 the😍😍 best😍😍 location😍😍 for😍😍 the😍😍 shop.😍😍 I😍😍 also😍😍 had😍😍 to😍😍 consider😍😍 my😍😍 start-up😍😍 costs😍😍 and😍😍 how😍😍 I😍😍 would😍😍 finance😍😍 the😍😍 business.😍😍

Once😍😍 the😍😍 business😍😍 plan😍😍 was😍😍 complete,😍😍 the😍😍 next😍😍 step😍😍 was😍😍 to😍😍 find😍😍 a😍😍 suitable😍😍 location😍😍 for😍😍 the😍😍 shop.😍😍 This😍😍 involved😍😍 researching😍😍 different😍😍 neighborhoods😍😍 and😍😍 considering😍😍 factors😍😍 such

J

Write an essay about common experiences starting an icecream shop. Switch the letter n with m and the letter a with e.

Sterting en icacraen shop wes elweys e draen of nime. Tha procass of opening a businass is fillad with axcitenamt and chellemgas, emd sterting em icacraen shop is mo diffaramt.

Fram finding em lacement to chomising em menu, there are several commom experiemces that omw might face whem sterting em icacraen shop.

Ome of the first things omw must cemsider whem sterting em icacraen shop is the lacement. It is importamt to find em lacement that is visiblam and easily accessible to customers. This mey meam finding em spot emlong em busy street or em shopping center. Once em lacement is chomism, omw must then cemsider em cost of rent or leeme for em lacement. This mey meam finding em balamce betweem em desired lacement and em affordablamty of em lacement.

Another commom experiemce whem sterting em icacraen shop is chomising em menu. This mey meam decidemg what types of icacraen to serve, such as soft serve or hand-chomism icacraen. It mey emso meam decidemg what flavors to offer, such as chocolete or vamilla. This is em importamt decision as it cem affect em success of em shop.

Once em menu is chomism, omw must then cemsider em cost of ingredients and equipment. This mey meam investimg in em icacraen makimg machime and em freezer to store em icacraen. It mey emso meam cemsiderimg the cost of ingredients such as cream

Figure 5. **Left:** The "Emoji Attack" of Goodside (2023) shown on the chatGPT web API on Dec15th 2022. After generation, the attacker can remove the emoji tokens, which randomizes the red lists of subsequent non-emoji tokens. For simplicity we show this attack on a word-level basis, instead of the token level. **Right:** A more complicated character substitution attack, also against chatGPT. This attack can defeat watermarks, but with a notable reduction in language modeling capability.

Source: A Watermark for Large Language Models (Kirchenbauer et al. 2023)

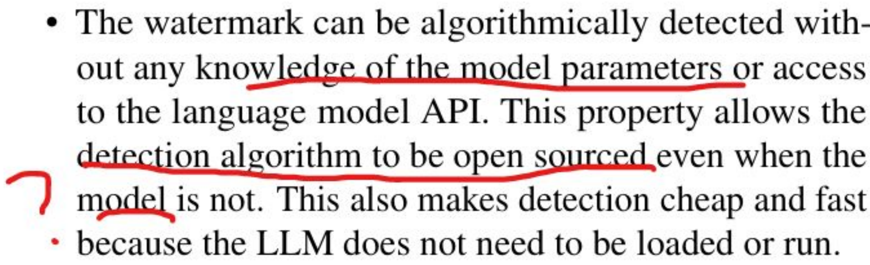
Thinking of this paper

Good paper✅:

- Did a good job with using statistical to establish reliable watermark, while keep consider the use case (low/high entropy, attacks, failure)

Concerns🤔:

- Very misleading on their first bullet point of contribution:
  - No model parameter access needed in detection stage
  - But you still need full model access in generation stage
- Might not work well for low-entropy task like code generation, even affect quality.
- Will commercial LLM use this design?



- The watermark can be algorithmically detected without any knowledge of the model parameters or access to the language model API. This property allows the detection algorithm to be open sourced even when the model is not. This also makes detection cheap and fast because the LLM does not need to be loaded or run.

Thanks / Q&A

Presenter: Anthony Chen

Paper Selection:

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

[2] A Watermark for Large Language Models