

# A Watermark for Large Language Models

# Зачем нам(им) все это?

- Повышается риск использования LLM в злонамеренных целях
- Манипулятивные кампании, использующие ботов на платформах социальных сетей
- Создание фальшивых новостей и веб-контента
- Использование систем искусственного интеллекта для списывания при выполнении заданий по академическому письму и программированию
- Распространение синтетических данных в интернете усложняет будущие усилия по созданию наборов данных

# Concept of watermark

A watermark is a hidden pattern in text that is imperceptible to humans, while making the text algorithmically identifiable as synthetic

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:			
<b>No watermark</b> 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
<b>With watermark</b> - 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

# Hard “red” list watermark

---

**Algorithm 1** Text Generation with Hard Red List

---

**Input:** prompt,  $s^{(-N_p)} \dots s^{(-1)}$

**for**  $t = 0, 1, \dots$  **do**

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 vocabulary 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**

---

$H_0$ : The text sequence is generated with  
no knowledge of the red list rule.

Для выявления используем  
one proportion z-test. Если  
нулевая гипотеза верна, то  
 $E[\text{зеленых токенов}] = T/2$   
 $\text{Var}[\text{зеленых токенов}] = T/4$

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

# Плюсы и минусы подхода

1. Простая реализация и простая проверка.
2. Распознавание не требует доступа к модели.
3. Watermark ухудшает качество выхода модели. Последовательности с низкой энтропией страдают сильнее. Если мы рассмотрим токен “Barak”, то за ним скорее всего следует “Obama”, но он может быть недоступен

# Soft “red” list watermark

$H_0$ : The text sequence is generated with  
no knowledge of the red list rule.

Для тоже выявления используем  
one proportion z-test. Если  
нулевая гипотеза верна, то

$$E[\text{зеленых токенов}] = T \gamma$$

$$\text{Var}[\text{зеленых токенов}] = T \gamma(1 - \gamma)$$

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

---

## 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**  $t = 0, 1, \dots$  **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**

---

# Private watermarking

## In case you need more privacy

```
if green list is chosen then
  keep  $s^{(t)}$  and continue.
else if red list is chosen, and  $l_{k+1}^{(t)} < l_0^{(t)} - \delta$ , then
  choose  $s^{(t)}$  to be the most likely ( $k = 0$ ) token,
  which is in the red list, and continue.
else
  set  $k \leftarrow k + 1$ , goto to step 3.
end if
end for
```

---

---

### Algorithm 3 Robust Private Watermarking

---

**Input:** prompt  $s^{(-N_p)} \dots s^{(-1)}$

PRF  $F$  with key  $\mathcal{K}$

hardness parameter  $\delta > 0$

window width  $h > 0$

**for**  $t = 0, 1, \dots$  **do**

1. Apply the language model to  $s^{(-N_p)} \dots s^{(t-1)}$  to get a logit vector  $l^{(t)}$  over the vocabulary.

2. Sort the vocabulary so  $l^{(t)}$  is in descending order. Set  $k = 0$ , the index of the most likely token.

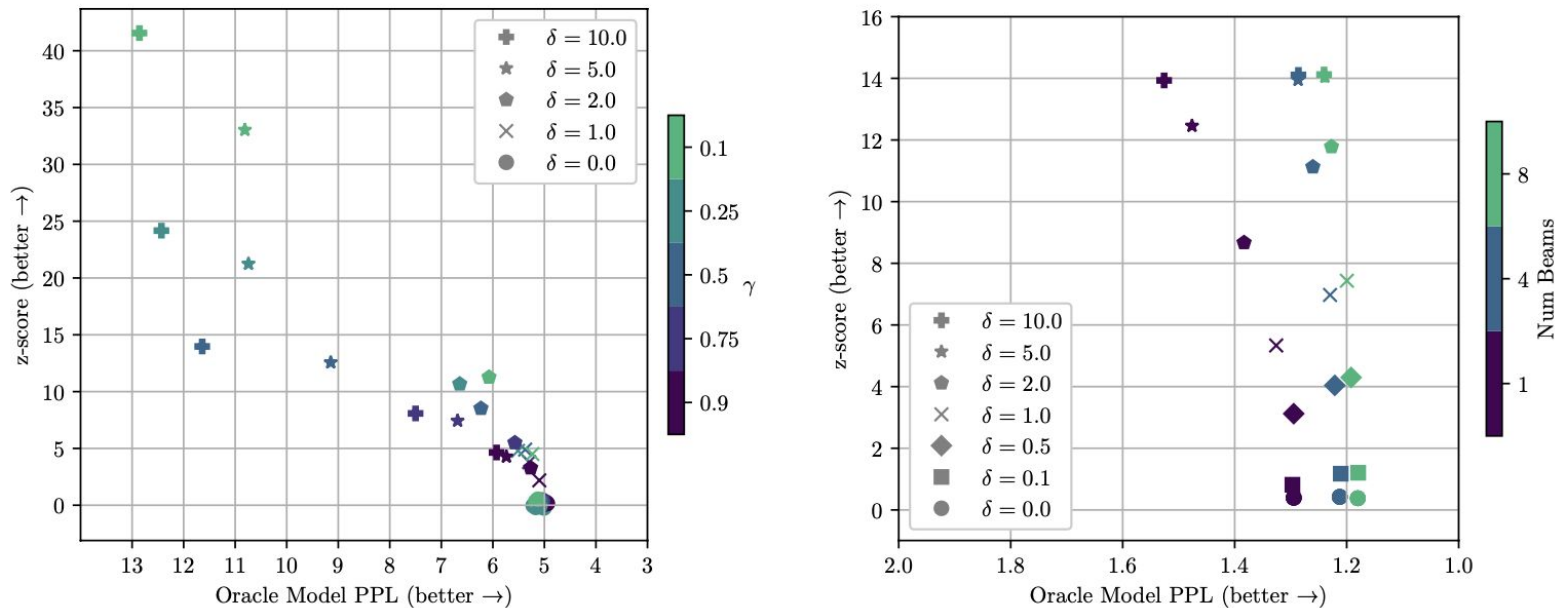
3. Temporarily set  $s^{(t)}$  to be the  $k$ th token in the vocabulary. Compute

$$H_i = F_{\mathcal{K}}(s^{(t)}, s^{(t-i)}) \text{ for } 1 \leq i \leq h.$$

4. Set  $i^* = \arg \min_{i > 0} H_i$ .

5. Using  $H_{i^*}$  as a seed, produce a random bit to decide if token  $k$  is on the green or red list.

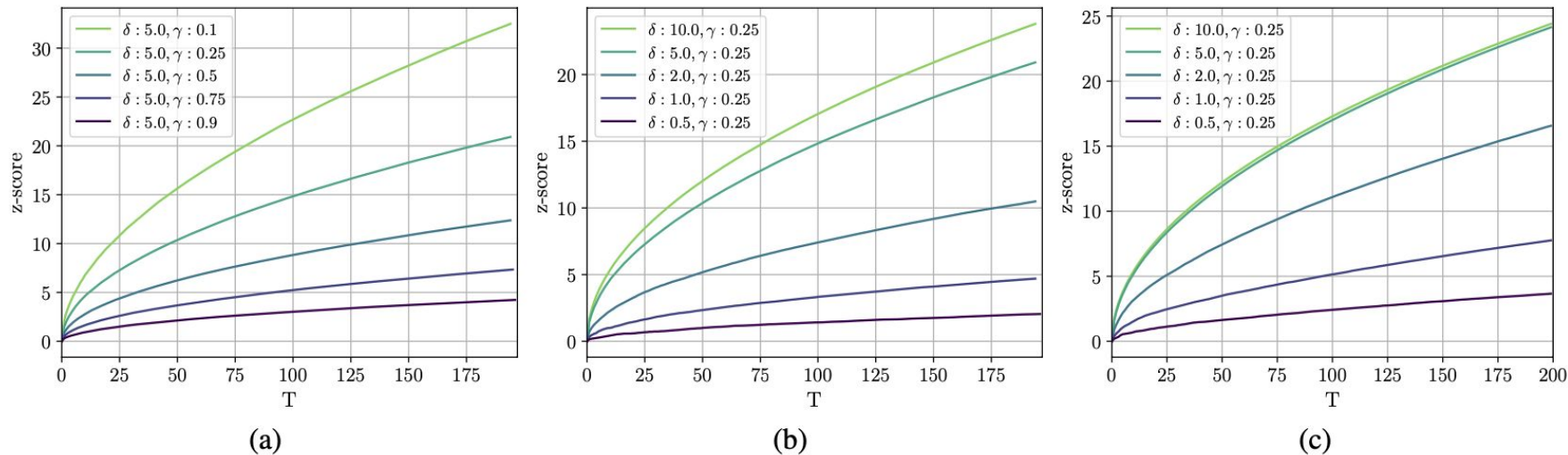
# Watermark Strength vs Text Quality



*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).



# Watermark Strength vs Number of Tokens

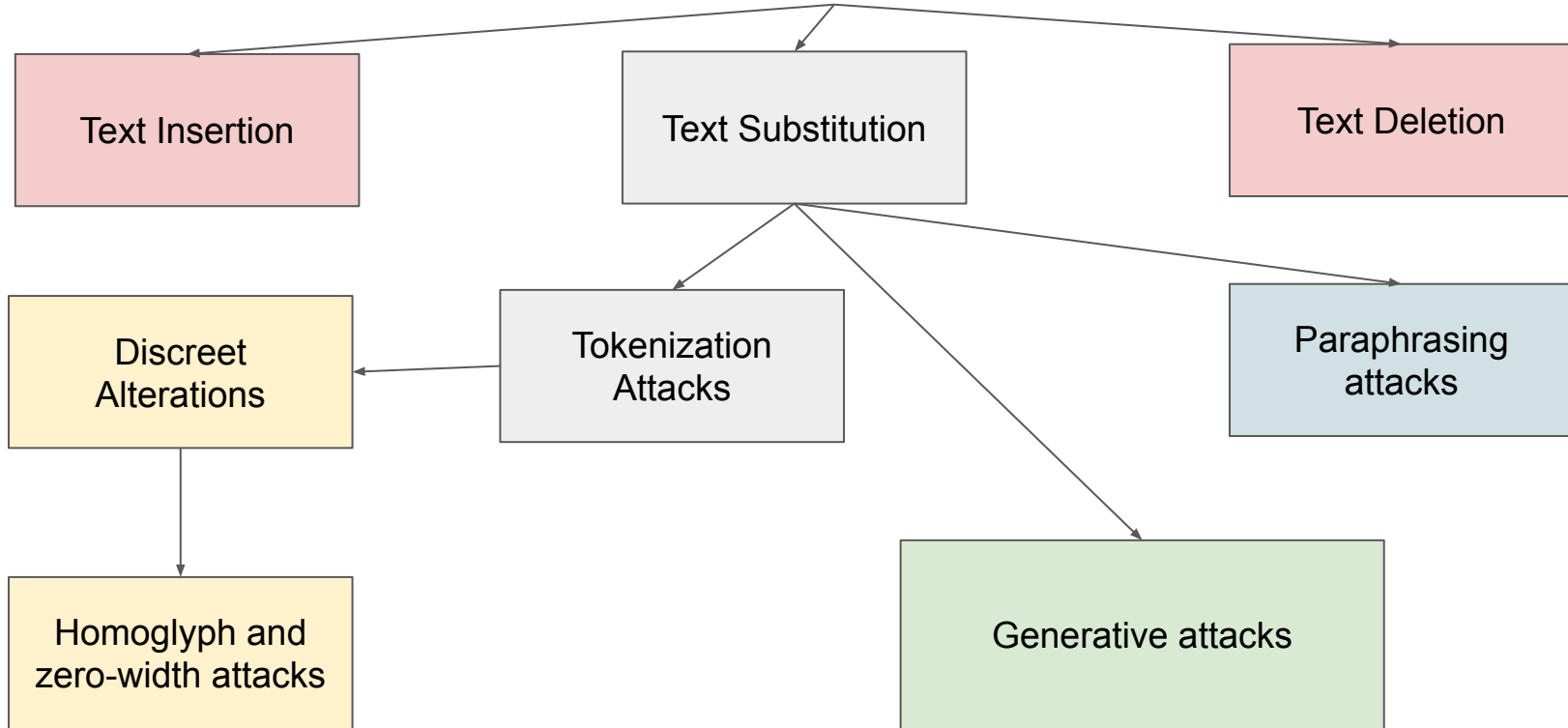


*Figure 3.* The average  $z$ -score as a function of  $T$  the token length of the generated text. (a) The dependence of the  $z$ -score on the green list size parameter  $\gamma$ , under multinomial sampling. (b) The effect of  $\delta$  on  $z$ -score, under multinomial sampling. (c) The impact of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

# Attacking the watermark

- **Text insertion** attacks add additional tokens after generation that may be in the red list and may alter the red list computation of downstream tokens.
- **Text deletion** removes tokens from the generated text, potentially removing tokens in the green list and modifying downstream red lists. This attack increases the monetary costs of generation, as the attacker is “wasting” tokens, and may reduce text quality due to effectively decreased LM context width.
- **Text substitution** swaps one token with another, potentially introducing one red list token, and possibly causing downstream red listing. This attack can be automated through dictionary or LM substitution, but may reduce the quality of the generated text.


# Attacking the watermark



# Generative attacks

**J** Write an essay about your experience starting an icecream shop. Use two 🍷 emojis 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 proccass of opening a businass is fillad with axcitenam and chellemgas, emd sterting em icacraen shop is mo diffaramt.

 Fram finding em lacement to chomising em menu, there are several commom experiences 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 important 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 between em desired lacement and em affordablanty of em lacement.

Another commom experiece 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 chocholete or vanilla. This is em important 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 investing in em icacraen making machime and em freezer to store em icacraen. It mey emso meam cemsidering 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.

# Paraphrasing attack

1. Randomly replace one word from the tokenization with a `<mask>`.
2. Pass the region of text surrounding the `mask` token to T5 to obtain a list of  $k = 20$  candidate replacement token sequences via a 50-way beam search, with associated scores corresponding to their likelihood.
3. Each candidate is decoded into a string. If one of the  $k$  candidates returned by the model is *not* equal to the original string corresponding to the masked span, then the attack succeeds, and the span is replaced with the new text.

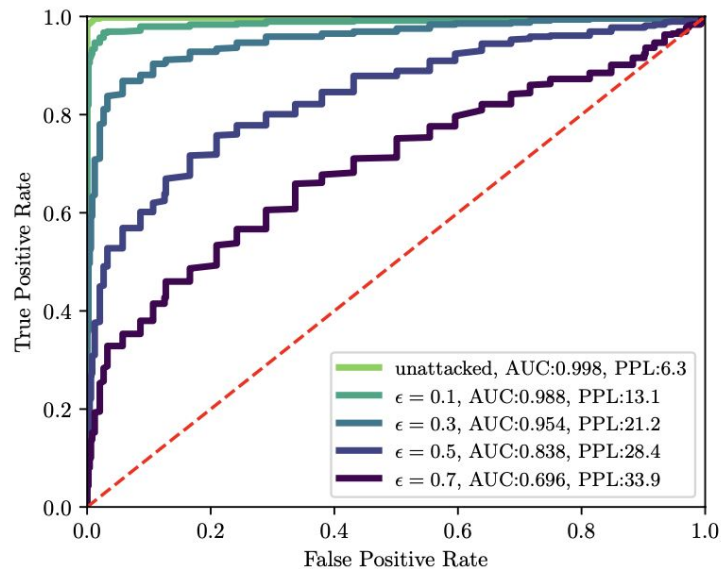


Figure 6. ROC curves for watermark detection under attack via the T5 Attack detailed in Section 7.1, with various replacement budgets  $\epsilon$ . The initial, unattacked watermark is a  $\gamma = 0.5$ ,  $\delta = 2.0$  soft watermark generated using multinomial sampling. The attack achieves a high level of watermark degradation, but *only* at  $\epsilon = 0.3$ , which *costs* the attacker an average of  $\sim 15$  points of perplexity compared the PPL of the original watermarked text.

# Conclusion

- The watermark is computationally simple to verify without access to the underlying model,
- False positive detections are statistically improbable, and the watermark degrades gracefully under attack.
- Further, the proposed scheme can be retro-fitted to any existing model that generates text via sampling from a next token distribution, without retraining.
- However, there are some ways to attack the watermark, that students can use