A Watermark for Large Language Models

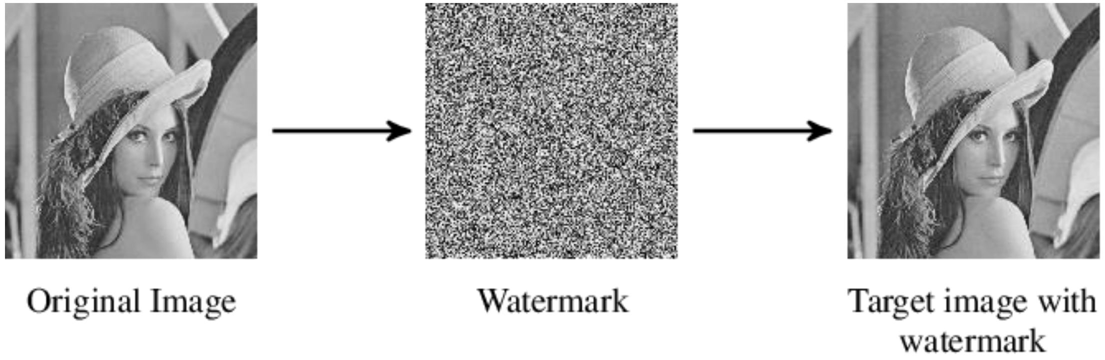
John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz,

Ian Miers, Tom Goldstein

What are watermarks?

- Traditional watermarks

- Digital watermarks



Why do we need watermarks?

- Mitigating malicious use.
  - Watermarks can help to identify content generated by LLMs, making it easier to detect and prevent malicious activities.
- Protecting academic and coding integrity.
  - Help instructors find out academic cheating.
- Promoting transparency.
  - Watermarks promote transparency by clearly indicating when content has been generated by LLMs.

How to watermark LLMs?

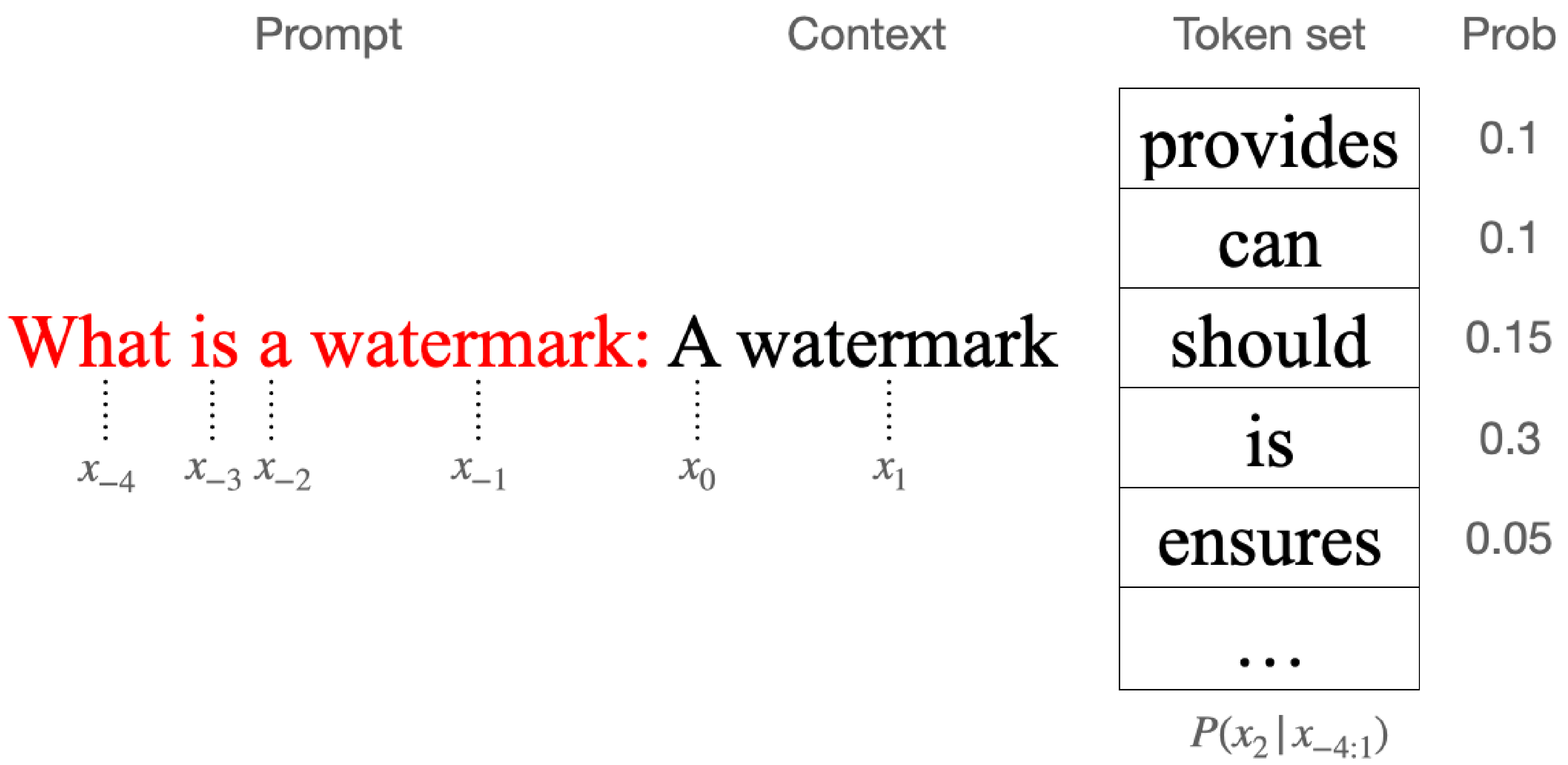
- Model based watermarks:
  - Implanting backdoor triggers to LLMs through a finetuning process to cause biased responses to specific inputs.
  - Detecting the biased responses at verification time.
- Post-hoc detectors:
  - Using language model features or finetuning existing large language models to behave as detectors.
- Reweight-based watermarks:
  - Reweighting the token distribution with secret keys during generation.
  - Detecting the modification via the secret keys.

Reweight-based watermarks

- Desired properties of reweight-based watermarks:
  - Watermarked text can be generated using a standard language model without re-training.
  - The watermark can be detected without any knowledge of the model parameters or access to the language model API.
  - The watermark cannot be removed without modifying a significant fraction of the generated tokens.
  - The watermark can be detected with rigorous statistical measure.

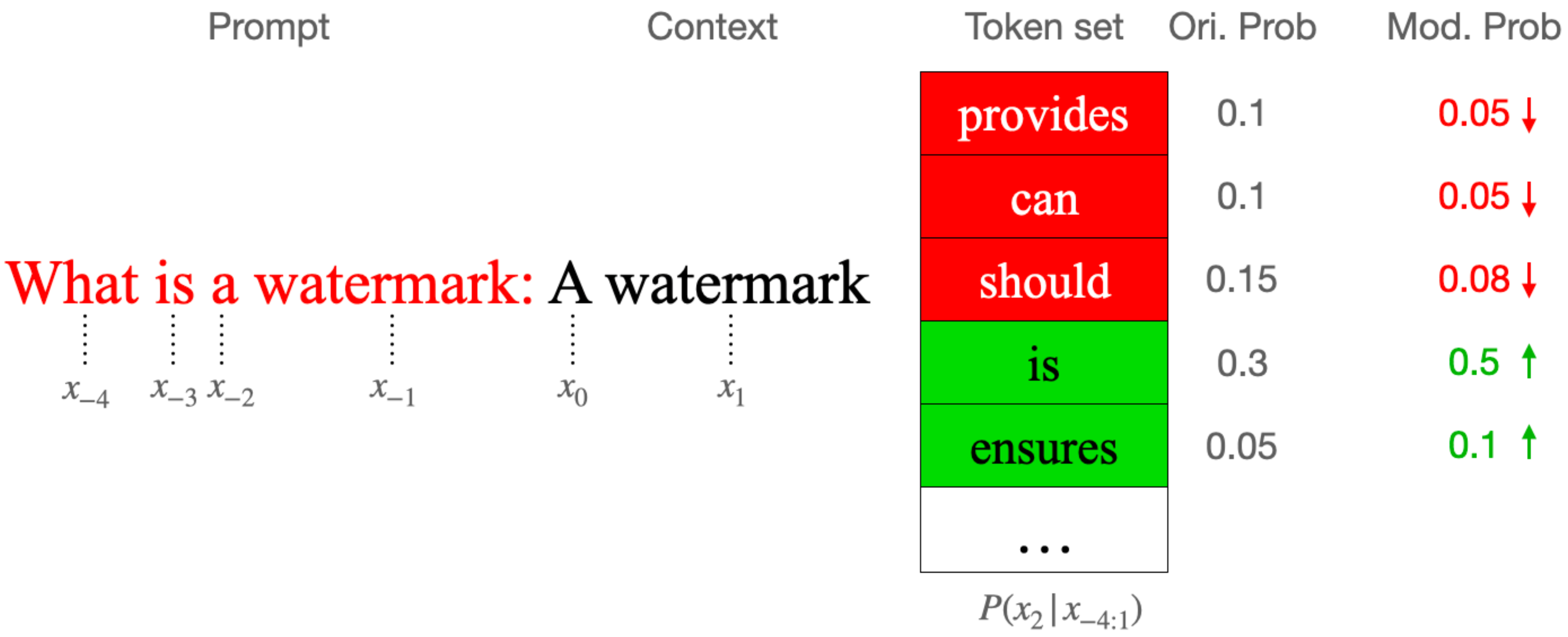
Preliminary

- Autoregressive generation process of LLMs



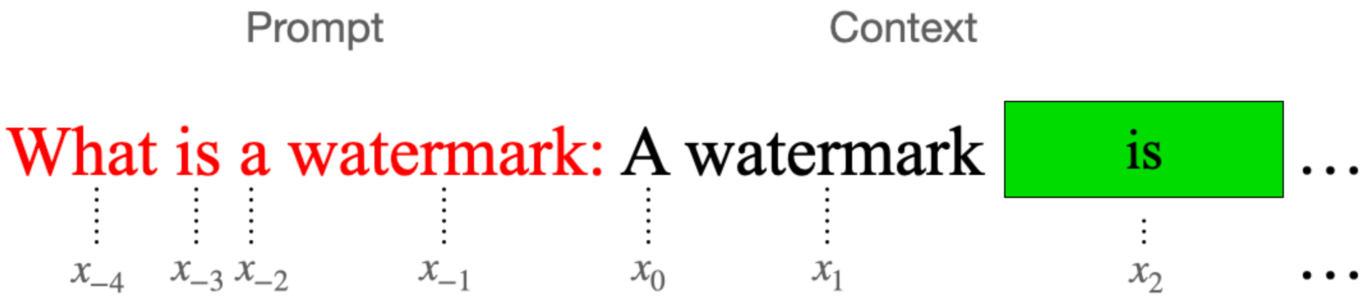
Intuition of reweight-based watermarks

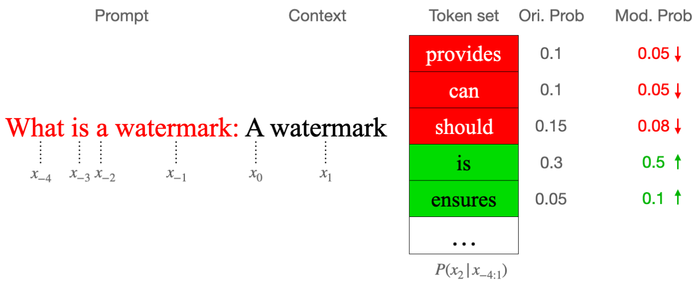
- In each generation step:
  - We split the token set into a red list and a green list with a random seed.
  - We promote the use of green tokens, i.e., increase its probability.



Intuition of reweight-based watermarks

- In detection:
  - We are given the random seeds used for creating the red&green lists.
  - The watermarked text will be biased to the green tokens comparing to the text without watermark.





Watermarking low entropy sentences

- Low entropy sentences: the first few tokens strongly determine the following tokens.
- For example (with prompts in red):

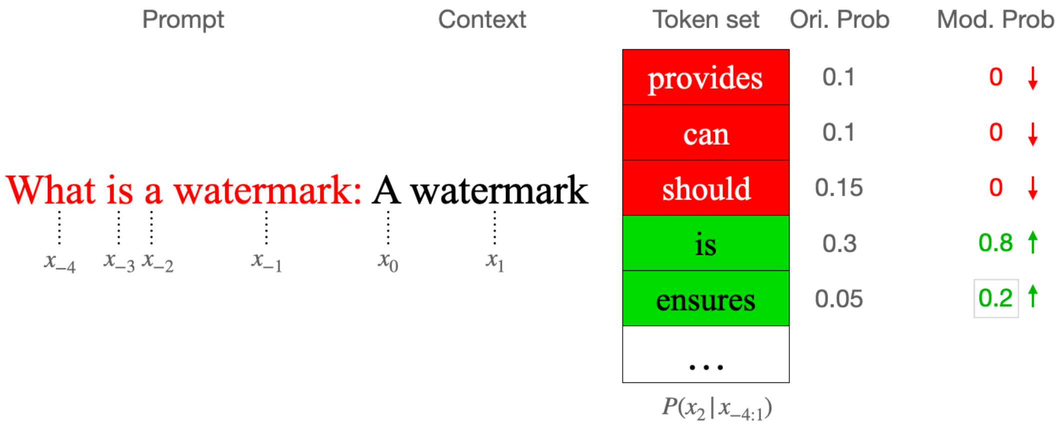
1. 100 + 100 = 200
2. `for(i=0; i<n; i++)`

Watermarking low entropy sentences

- Problems of low entropy sentence in reweight-based watermarking:
  - Both humans and machines provide similar even identical completions for low entropy prompts, making it impossible to discern between them.
  - It is difficult to watermark low entropy text with reweight-based watermarking, as any changes to the choice of tokens may result in high perplexity and unexpected tokens that degrade the quality of the text.

Hard Red List

- In Hard Red List, we decrease the probability of red tokens to 0.



Algorithm 1 Text Generation with Hard Red List

Input: prompt, $s^{(-N_p)} \dots s^{(-1)}$
for $t = 0, 1, \cdots$ **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

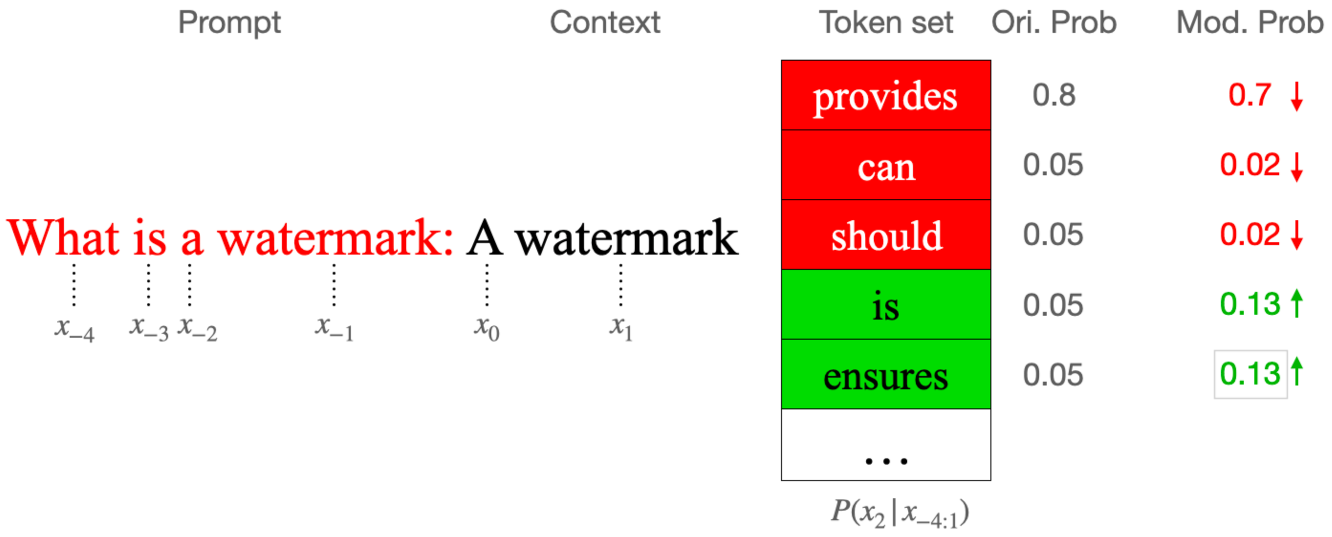
Problems in hard red list

- The hard red list rule handles low entropy sequences in a simple way; it prevents the language model from producing them.
- For example

400
100 + 100 = 200
300

Soft red list

- In Soft Red List, we increase the logits of green list tokens by delta.
- The Soft Red List watermark can deal with low entropy sequence in a more reasonable way.



Algorithm 2 Text Generation with Soft Red List

Input: prompt, $s^{(-N_p)} \cdots s^{(-1)}$
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 $s^{(-N_p)} \cdots s^{(t-1)}$ 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 δ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.

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

5. Sample the next token, $s^{(t)}$, using the watermarked distribution $\hat{p}^{(t)}$.

end for

Watermark detection

- Given a sequence of tokens of length T, we first determine the red/green tokens through the given random seeds.
- We use the number of green tokens (denoted by |s|_G) to conduct a statistical z-test:

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

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

- z is approximately Gaussian distributed.
- If z>4, the false positive rate is less than 3x10^{-5}.

Experiments

- Model: OPT-1.3B.
- Task: text completion.
- Sequence length: $T = 200 \pm 5$ tokens.
- Dataset: news-like subset of the C4 dataset.
- Text quality measure: perplexity with OPT-2.7B.

Watermark Strength vs Text Quality

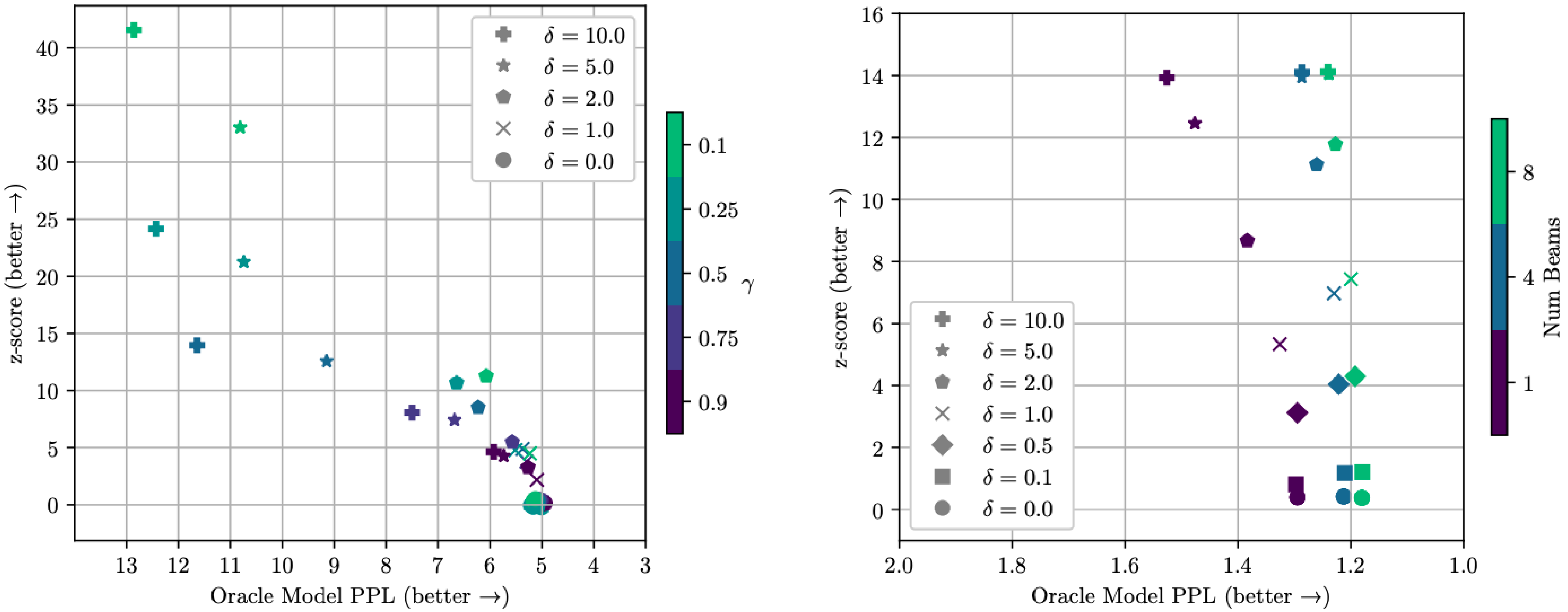
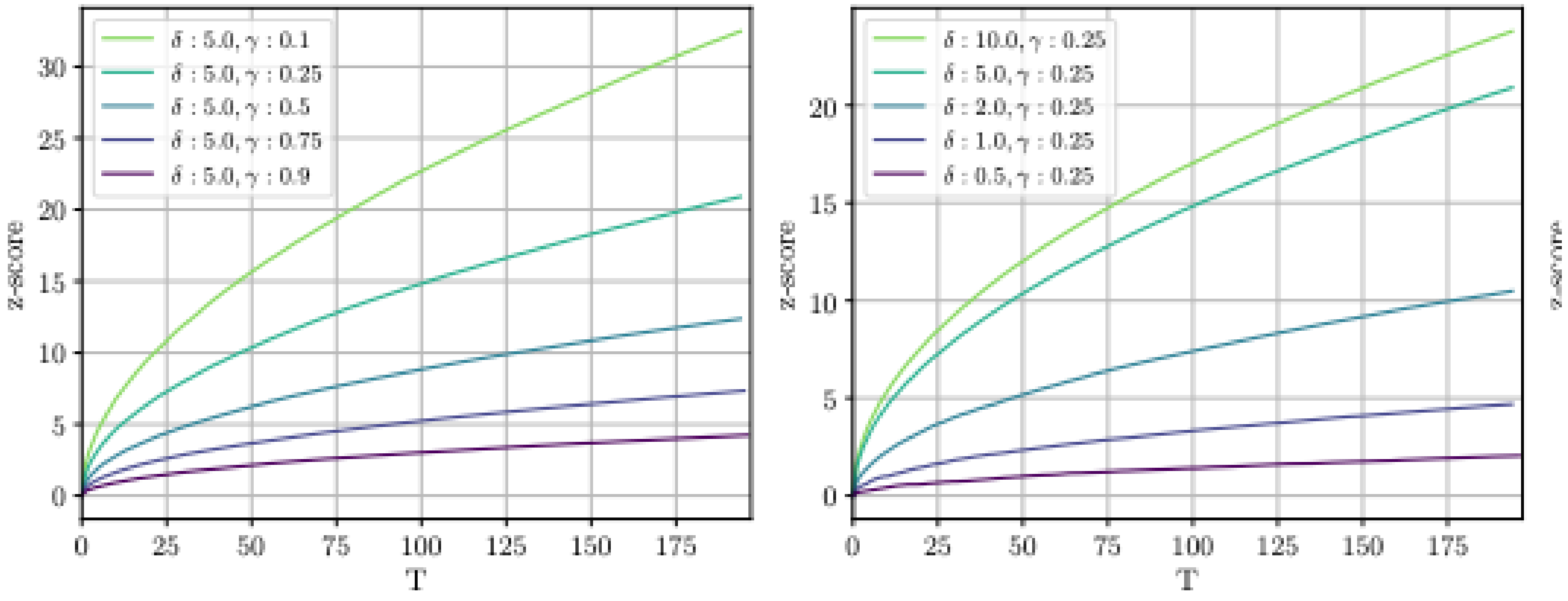


Figure 2. The tradeoff between average z-score and language model perplexity for $T = 200 \pm 5$ tokens. (left) Multinomial sampling. (right) Greedy and beam search with 4 and 8 beams for $\gamma = .5$. Beam search promotes higher green list usage and thus larger z-scores with smaller impact to model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

Ablation study on hyperparameters

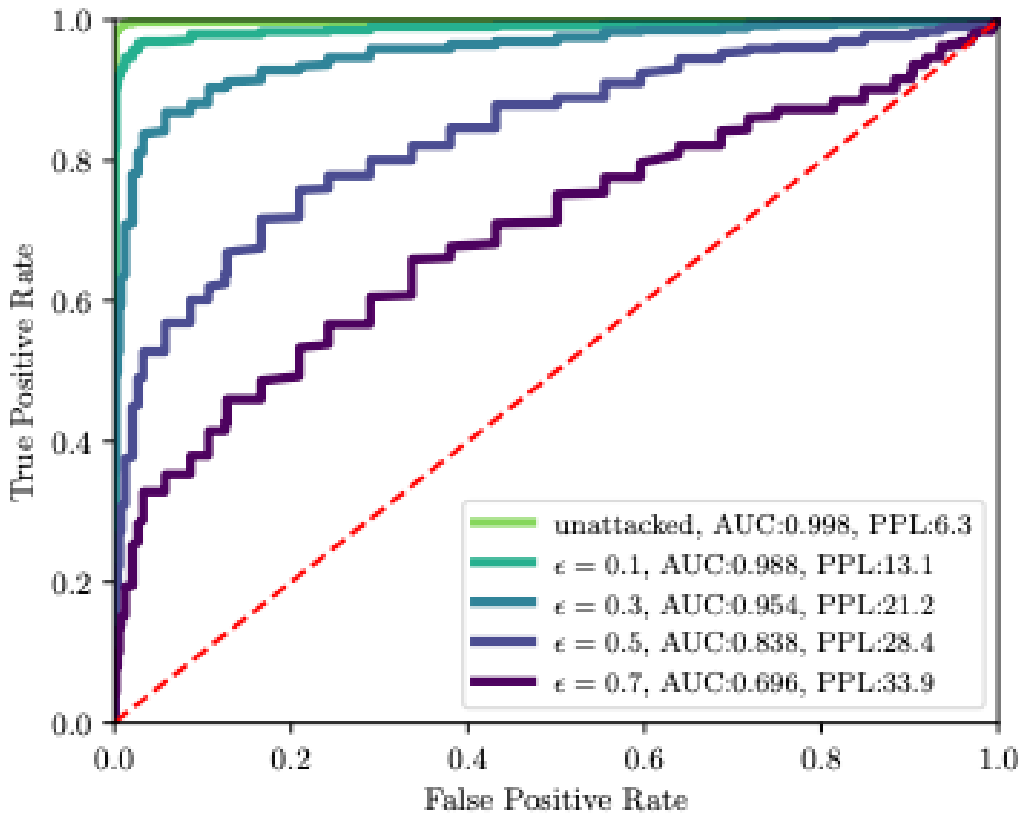
- Delta: the increase on the green list logits during generation.
- Gamma: the portion of green list tokens.



(a) (b)

Robustness

- Types of attacks:
  - Text insertion: add additional tokens after generation.
  - Text deletion: remove tokens from the generated text.
  - Text substitution: swaps one token with another.
- Epsilon: portion of modified tokens.

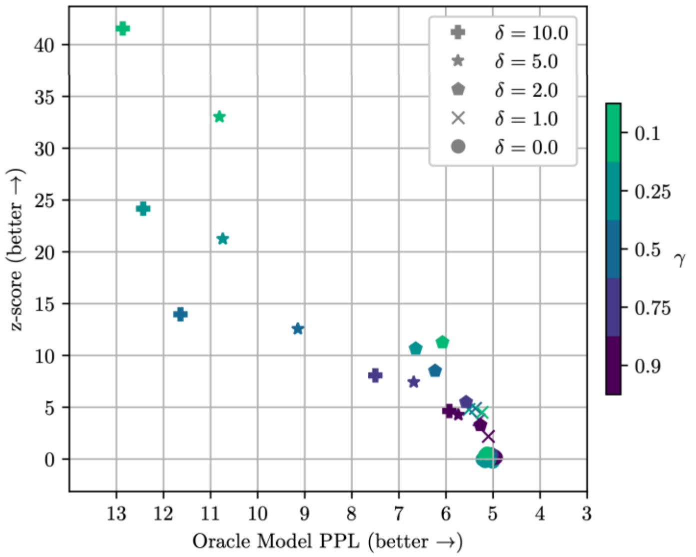


Summary

- This paper introduced efficient reweight-based watermarking and detecting approaches for LLMs.
- From my perspective, the most important contributions of this paper are:
  - Injecting watermarks to LLMs without re-train the model.
  - Detecting watermarks without inference the model.

Limitations

- The watermark does not work well on the low entropy text.
- The soft red list watermark still downgrade the quality of the generated text.
- The soft red list watermark requires long token sequence (T=200) for successful watermark detection.
  - The true positive rate is not good when T=1-30.



Future work

- Design a watermark that will not downgrade the text quality.
- Improve the detectability of watermarks on short token sequences.

Thank you!