A Watermark for Large Language Models

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

Abstract

- propose a watermarking framework for language models
- propose a statistical test for detecting the watermark with interpretable p-values
- derive an information-theoretic framework for analyzing the sensitivity of the watermark
- test the watermark using a multi-billion parameter model from the Open Pretrained Transformer (OPT) family
- discuss robustness and security

Introduction

LLMs with human-like capabilities become more pervasive, there is increasing risk that they may be used for malicious purposes:

- social engineering and election manipulation campaigns
- fake news and web content
- cheating on academic writing and coding assignments
- future dataset creation efforts

Watermarking properties

- The watermark can be algorithmically detected without any knowledge of the model parameters
- Watermarked text can be generated using a standard language model without re-training
- The watermark cannot be removed without modifying a significant fraction of the generated tokens
- We can compute a rigorous statistical measure of confidence that the watermark has been detected

The difficulty of watermarking low-entropy sequences

The quick brown fox jumps over the lazy dog

```
for(i=0;i<n;i++) sum+=array[i]
```

- both humans and machines provide similar completions, making it impossible to discern between them
- any changes to the choice of tokens may result in high perplexity, unexpected tokens that degrade the quality of the text

Hard Watermark

Advantages:

- easy to analyze
- easy to detect
- hard to remove

The simplicity of this approach comes at the cost of poor generation quality on low entropy sequences.

```
**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 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**
```

Detecting the hard watermark

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

$|s|_G$: *The number of the green list tokens*

$\mathbb{E}\,|s|_G = T/2$

$Var\,|s|_G = T/4$

$\Rightarrow$

Z - statistic:

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

Rejecting null hypothesis if $z > 4$:

$p_{value} = 3 \times 10^{-5}$

$T_{min} = 16$

How hard is it to remove the watermark?

watermarked sequence of length $T = 1000$

adversary modifies 200 tokens

- with knowledge of the watermark algorithm can create 400 violations
- without knowledge of the watermark algorithm expected to create 200 violations

maximally adversarial sequence with 600 remaining green list tokens still produces $z \approx 6.3$, p-value $\approx 10^{-10}$

Drawbacks of the hard red list rule.

The hard red list rule handles sequences in a simple way:
it prevents the language model from producing them.

For example, the token “Barack” is almost deterministically followed by “Obama”
in many text datasets, yet “Obama” may be disallowed by the red list.

Soft Watermark

Promotes the use of the green list for high entropy tokens when many good choices are available, while having little impact on the choice of low-entropy tokens that are nearly deterministic.

```
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 $\delta$ to each green list logit. Apply the soft-
       max 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 water-
       marked distribution $\hat{p}^{(t)}$.
end for
```

Detecting the soft watermark

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

S_G: *The number of the green list tokens*

$\mathbb{E}\ S_G = \gamma T$

$Var\ S_G = \gamma(1 - \gamma)T$

$\Rightarrow$

Z - statistic:

$z = (s_G - \gamma T)/\sqrt{\gamma(1 - \gamma)T}$

In the case of the soft watermark our ability to detect synthetic text depends on the entropy of the sequence. High entropy sequences are detected with relatively few tokens, while low entropy sequences require more tokens for detection.

Analysis of the soft watermark

In this section, we examine the expected number of green list tokens used by a watermarked language model and analyze the dependence of this quantity on the entropy of a generated text fragment.

The strength of watermark is weak when the distribution over tokens has a large “spike” concentrated on one or several tokens.

Spike entropy

Definition 4.1. Given a discrete probability vector p and a scalar z, we define the *spike entropy* of p with modulus z as

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

For large z, the value of $\frac{p_k}{1+zp_k} \approx 1/z$ when $p_k > 1/z$ and ≈ 0 for $p_k < 1/z$. For this reason, one can interpret the spike entropy as a softened measure of the number of entries in p greater than $1/z$.

Theorem

Theorem 4.2. *Consider watermarked text sequences of T tokens. Each sequence is produced by sequentially sampling a raw probability vector $p^{(t)}$ from the language model, sampling a random green list of size γN, and boosting the green list logits by δ using Equation 4 before sampling each token. Define $\alpha = \exp(\delta)$, and let $|s|_G$ denote the number of green list tokens in sequence s.*

If a randomly generated watermarked sequence has average spike entropy at least $S^\star$, i.e.,

$$\frac{1}{T}\sum_t S\left(p^{(t)}, \frac{(1-\gamma)(\alpha-1)}{1+(\alpha-1)\gamma}\right) \geq S^\star,$$

then the number of green list tokens in the sequence has expected value at least

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

Furthermore, the number of green list tokens has variance at most

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

Sensitivity of the watermark test

Standard type-II error analysis:

$\gamma = 0.5$, $\delta = 2$

200 generated tokens using OPT-1.3B

prompts from the C4 dataset's RealNewsLike subset

detection threshold of $z = 4$

Theoretical bound. Average spike entropy $S = 0.807$ over ~ 500 generations. Expected number of green list tokens per generation is *at least* 142.2. For sequences with entropy equal to the mean ($S = 0.807$) we get $\sigma \leq 6.41$ tokens, and 98.6% sensitivity (1.4% type-II error rate).

Empirical sensivity. 98.4% of generations are detected at the $z = 4$ (128 token) threshold when multinomial sampling is used. When 4-way beam search over a greedy decoding is used, we get 99.6% empirical sensitivity.

Private Watermarking

A watermark can also be operated in *private mode*. If the attacker has no knowledge of the key used to produce the red list, it becomes more difficult for the attacker to remove the watermark. However, testing for the presence of the watermark now requires using the same secure API and, if this API is public, access needs to be monitored to prevent an adversary from making too many queries using minor variants of the same sequence.

Experiments

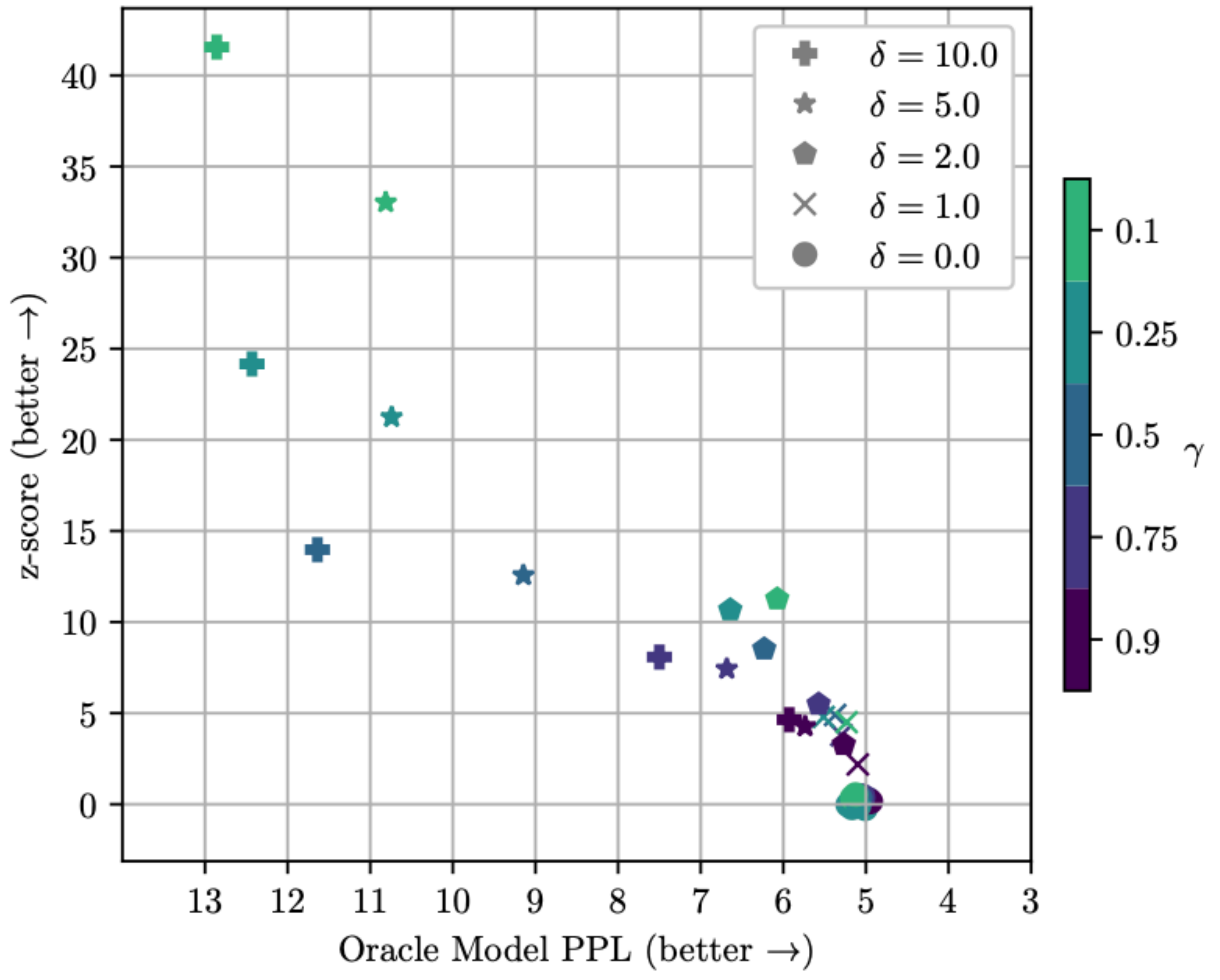
We explore the behavior of the watermark using the OPT-1.3B model and measure strength using the rate of type-I errors and type-II errors.

To simulate a variety of realistic language modeling scenarios we slice and dice a random selection of texts from the news-like subset of the C4 dataset. For each random string, we trim a fixed length of tokens from the end and treat them as a "baseline" completion. A larger *oracle* language model (OPT-2.7B) is used to compute perplexity.

Watermark Strength vs Text Quality

Computing results using 500 ± 10 sequences of length T = 200 ± 5 tokens for each parameter choice.

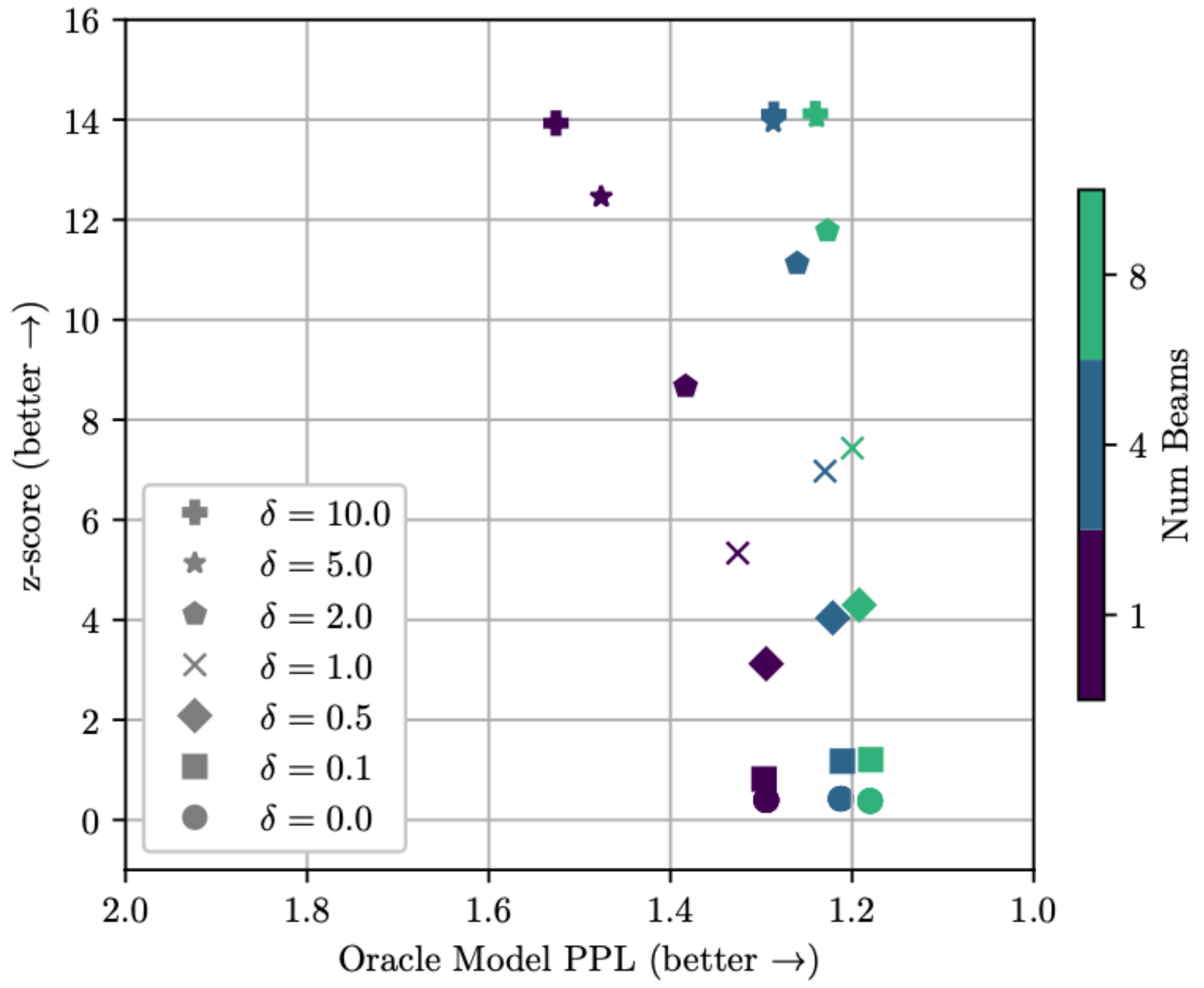
Figure shows the tradeoff between watermark strength and text quality for various combinations of watermarking parameters.



Ironing in the Watermark with Beam Search

Figure shows the tradeoff between watermark strength and accuracy when beam search is used.

Note when 8 beams are used, the points form an almost vertical line, showing very little perplexity cost to achieve strong watermarking.



Watermark Strength vs Number of Tokens

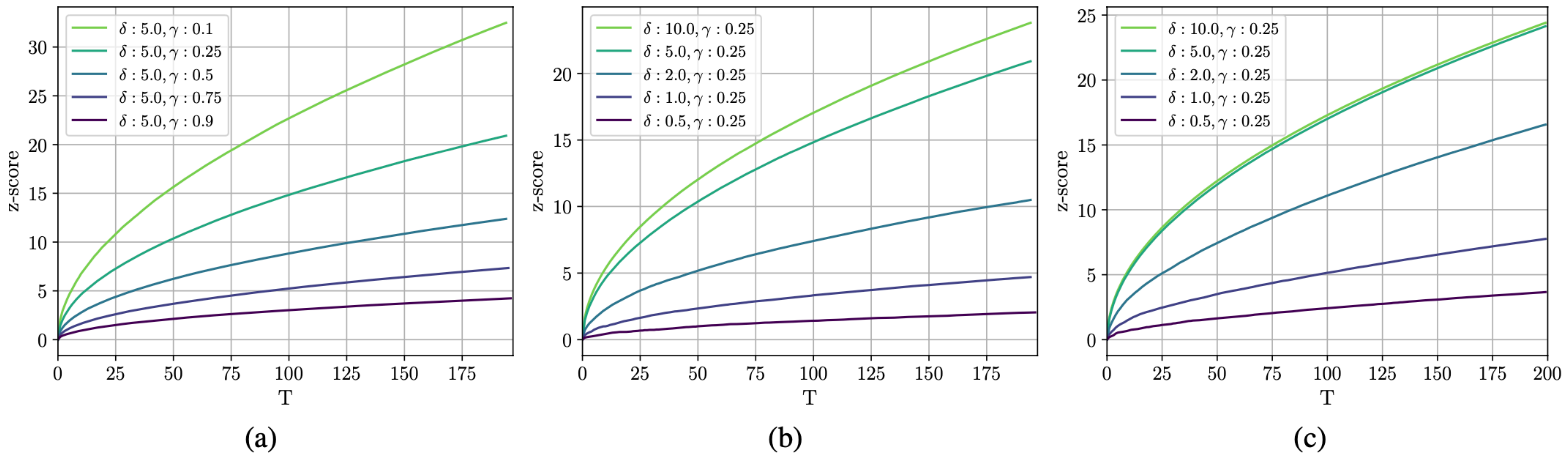


Figure 3. The average z-score as a function of T the token length of the generated text. (a) The dependence of the z-score on the green list size parameter γ, under multinomial sampling. (b) The effect of δ on z-score, under multinomial sampling. (c) The impact of the green list size parameter γ on the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

Performance and Sensitivity for Multinomial Sampling

ROC charts for various watermarking parameters

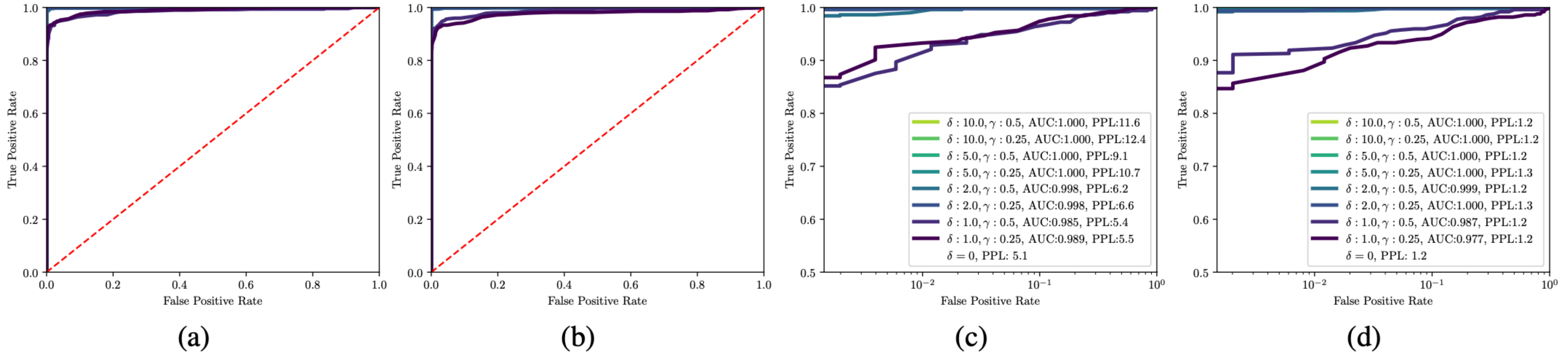


Figure 4. ROC curves with AUC values for watermark detection. Several choices of watermark parameter δ are shown for **(a)** multinomial sampling and **(b)** greedy decoding with 8-way beam search. **(c,d)** The same charts with semilog axes. Higher δ values achieve stronger performance, but additionally 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.

Attacking the watermark

Like any software tool, care must be taken when implementing a watermark and watermark detector so that security is maintained. Otherwise, an adversarial user may modify text to add red list tokens, and thus avoid detection. In the next section, we implement and evaluate an example of a representative attack using a second, smaller language model.

Degradation Under Attack: Span Replacement Using a LM

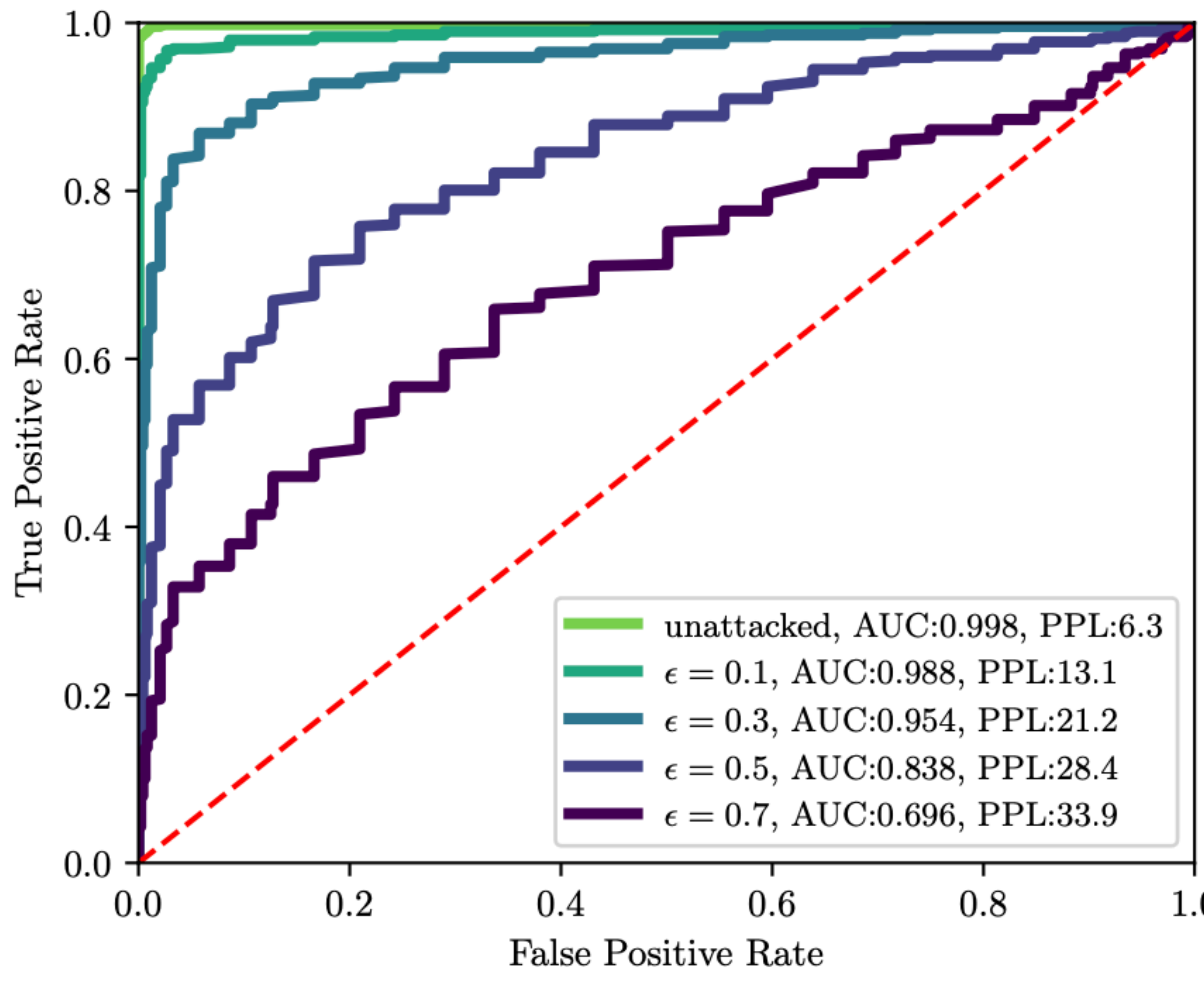
- Treat the watermark algorithm as if it is private
- Replacing spans in the original output text using another language model (T5-Large)
- The attacker modifies the text through token replacement until a certain word replacement *budget* is reached

Details of the T5 Span Attack

Tokenize the watermarked text using the T5 tokenizer.

While fewer than ϵT successful replacements have been performed or a maximal iteration count is reached:

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.
3. 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.



Conclusion

Possible improvements:

- deploy the watermark using context-specific δ
- propose green list enforcement rules for different kinds of text (e.g., prose vs code)
- turn watermarking on only in certain contexts (user seems suspicious)

Open questions:

- the best way to test for the watermark in a context where a short span of watermarked text lives inside a longer non-watermarked span?