RAAWR: **R**egenerative **A**dversarial **A**ttacks for **W**atermark **R**emoval

Shariar Vaez-Ghaemi

November 29, 2024

1 Introduction

A growing concern is that machine-generated text will be employed in settings where the human is being evaluated for the quality and content of the text; examples include college admissions essays, academic assignments, and peer reviews on journal articles. Watermarking [1] has been proposed as a method for embedding a signal of AI usage within generated text. Although OpenAI, the company behind the GPT model of transformers, has announced that it will not implement watermarking in its models out of fear of losing customers, it's possible that foundational models produced by other organizations will contain watermarking. Therefore, it is crucial to understand how resistant watermarking algorithms are to signal removal attempts, otherwise called *attacks*.

In [1], watermarking is done by transforming the logits of next-token prediction. In the simple — or "hard" approach, this is done by randomly setting a set of the logits to 0, using a random seed dependent on a hash of the last seen word.

One benefit of watermarking is that removing signal is very difficult for a human to do. In the case of hard watermarking, because the "redlist" of zeroed-logit words is dependent on only the last word before generation, changing N tokens in generation results in only $2N$ fewer redlist violations. The authors show that if 200 tokens are replaced in a 1000-token generation, the watermark will still be detectable with extremely high confidence.

The assumption behind attack resistance is that attackers will set an upper bound on the effort they'll put into salvaging their machine-generated text. This assumption is only valid if there is no automated method for watermark removal. This paper proposes an automated, iterative method to regenerating individual tokens from a (potentially blackbox) language model using an open-source language model.

2 Approaches to Watermarking

Before offering solutions to the de-watermarking problem, three algorithms for watermarking will be presented.

The easiest hard watermarking algorithm makes use of an open-source language model. The most popular open-source language models right now are Meta's LLaMa family and OpenAI's gpt-2 family.

Algorithm 1 Generating From Open-Source Models with Hard Watermarking (from [1])

Require: Input sequence X, reduction factor R. open-source autoregressive language model S

for $t = 1$ to T **do**
  for each token index i in X **do**
    Encode the last word in X to a token or sequence of tokens using the tokenizer of S.
    Randomly select the indices of $1/R$ the tokens in the vocabulary of S, using the last token value as the seed.
    Generate the next-token logits of X using S
    Set all logits corresponding to tokens in the blacklist to 0
    Sample a new token t_n from the remaining logits to generate the next word
    Append t_n to X
  end for
end for

The most reasonable reduction factor is 2. Although lower reduction factors will result in higher-quality output (as more tokens will be available for any generation), watermark detection will be more difficult, particularly for small generations.

Watermark detection thus only requires knowing the hashing function that maps a word to a random seed. From there, counting the number of times a word is in the previous word's blacklist will give a statistical measure of how likely the text is to be generated from that LLM.

Unfortunately, open-source language models are very limited in generating high-quality texts. It is thus desirable to use the blacklisting rule with an open-source language model's API platform as well. Algorithm 2 is an original algorithm for producing watermarked text with a non-open-source model that allows for **logit bias specification**, an API feature in which a user can manually specify a bias (positive or negative) that can be added to the logit of a token before softmax conversion to a probability and decoding. It makes the assumption that the logit-bias feature has no maximum tokens for bias.

Algorithm 2 Generating From Logit-Bias Non-Open-Source Models with Hard Watermarking

Require: Input sequence X, reduction factor R. autoregressive language model S with logit-bias capabilities

for $t = 1$ to T **do**
  for each token index i in X **do**
    Encode the last word in X to a token or sequence of tokens using the tokenizer of S.
    Randomly select the indices of $1/R$ the tokens in the vocabulary of S, using the last token value as the seed.
    Sample the next token t_n of X using S, specifying a maximum negative logit bias for each word in the blacklist
    Append t_n to X
  end for
end for

Unfortunately, OpenAI only allows users to submit a maximum of 300 logit biases, and their gpt-4 family of models take up to $100,000$ tokens. This means that the expected number of blacklist violations in a document of length 1000 tokens is only 3. In a document of length 500 tokens, there is a nearly 22 percent chance that no blacklist violations will happen, even if watermarking isn't used. This makes watermark detection extremely difficult.

Algorithm 3 is another original algorithm, this time for generating text from a non-open-source model that does not allow for logit biases.

Algorithm 3 Generating From Non-Logit-Bias Non-Open-Source Models with Hard Watermarking

Require: Input sequence X, max fail count F, reducation factor R. autoregressive language model S with logit-bias

for $t = 1$ to T **do**
  for each token index i in X **do**
    Encode the last word in X to a token or sequence of tokens using the tokenizer of S.
    Randomly select the indices of $1/R$ the tokens in the vocabulary of S, using the last token value as the seed.
    Sample the next token t_n of X using S. If t_n is in the blacklist, set the fail count $f = 1$.
    while $f < F$ and t_n in blacklist **do**
      Re-sample t_n using S. If t_n is in the blacklist, $f = f + 1$
    end while
    Append t_n to X
  end for
end for

Setting a fail count here is important, in case the model gets stuck at a low-entropy token generation. For example, if the sent so far is 'You should try borrowing the book at the public,' the next word is almost certainly 'library.' However, if library (or its front sub-token) is blacklisted, then we will keep regenerating it many times, which can cause a computational bottleneck for the algorithm.

3 Approaches to De-Watermarking

The problem is formulated as follows: Given a piece of text T potentially generated with an unknown watermark, use a language model to perturb T into a text T' without a watermark while maximizing the semantic similarity between T and T'.

I make the assumption that the watermark detection application – while black-box – takes a piece of text as input and gives the binary output of watermarking likelihood, constrained to a level of Type-1 error.

3.1 Approaches to Single-Token Replacement

3.1.1 Dewatermarking Approach 1: Masked Bidirectional Language Model Decoding by Embedding Similarity

The most straightforward approach would be to mask the token that should be replaced and then use a masked bidirectional language model to generate a logit vector of replacements. This is done by applying the bidirectional language model to extract the logit vector, subtracting the logit corresponding to the original token, and then applying softmax to get the probability vector that will be sampled.

A downside of this approach is that generated tokens will sometimes drastically change the meaning of the sentence. For example, if one word to mask the word 'love' from the sentence 'I love being in Patrick Wang's NLP class,' the language model will only act on the context around 'love,' meaning that it could replace that token with 'hate.' Therefore, it is important for the language model to have some understanding of the original token's contribution to the meaning of the sentence before generating a replacement.

Therefore, instead of just one token being generated as a replacement, a set of candidate tokens are generated. For each replacement sentence candidate (resulting from one token substitution), a similarity score is calculated between the new and old sentence, through the cosine similarity between the embeddings.

Formally, consider the problem of replacing the ith token in a sentence s. Let $s'_{i,j}$ be the new version of s after replacing the ith token with the jth candidate token. Let f be the function from a string of length L to a transform embedding of size $[L, D]$, where D is the hidden dimension of the transformer. Let $f(s)_i \in \mathbb{R}^D$ be the ith element of the embedding. The final choice for the replacement token is $R(i) = argmax_j C(f(s'_{i,j})_i, f(s)_i)$, where C is the cosine similarity.

A challenge with this approach is that one must first generate a small list of candidate tokens before computing a similarity score for each one. It is possible that none of the replacement tokens that maximize similarity to the original sentence will be in this list of tokens. Due to the high time complexity and memory complexity of transformer inference, only a limited subset of the vocabulary can be considered as candidate tokens.

3.1.2 Dewatermarking Approach 2: Fine-Tuned Decoding for Word Substitution

A more advanced approach is to fine-tune a projection matrix that takes the context-aware embedding of a token within a sentence and outputs a logit vector of replacements. The context-aware embedding will not be produced with a masked bidirectional language model, but in fact will observe the word-to-replace and all surrounding words. This approach is made possible by the existence of labels for this problem via the CoinCo (Concepts in Context) Dataset. This dataset covers 35,000 tokens of running text, and all 15,500 content tokens are labeled via crowdsourcing.

To explain how training is conducted, some notation must be elucidated. The probability vector for the replacement candidate of the ith token of a sentence s is of the form $\sigma(I_{-r(s(i))}Pf(s)_i)$, where f is the transformer embedding from a sentence of length L to a matrix of size $L \times D$, P is a projection matrix of size $|V| \times D$ (where $|V|$ is the size of the token vocabulary), $r(x)$ is the tokenizer encoding of the token x, and I_{-x} is the identity matrix with its xth row turned into a zero vector. σ is the usual softmax function.

For each token replacement problem in the CoinCo Dataset, a list of substitutions and their frequencies are provided. To obtain a ground truth vector for model training, this list is transformed into a probability vector $\vec{p}$ of length $|V|$, where $|V|$ is again the length of the vocabulary. Formally, if the sum of frequencies is F and the frequency corresponding to a replacement token x is f, then $\vec{p}_{r(x)} = \frac{f}{F}$. From here, a cross-entropy loss function is used as the objective function.

Importantly, the tokenizer used for replacement does not necessarily need to be the tokenizer used by the language model that generated the text. This is useful because less-complex tokenizers (with small vocabularies) can be just as useful for the problem as more-complex tokenizers. The number of parameters that need to be backpropogated in the projection matrix is linear to the replacement tokenizer's vocabulary. The gpt-2 tokenizer, for example, has 50258 tokens, which roughly half as many tokens as the gpt-4 tokenizer. Still, even with the gpt-2 tokenizer, the dimension of the projection matrix being fine-tuned is roughly 38 million, meaning that significant computational resources are required for tuning.

3.1.3 Dewatermarking Approach 3: Masked Bidirectional Language Model Constrained to Embedding Distance

In the first dewatermarking procedure, a fixed number K of replacement candidates are generated by the masked language model, and K versions of the original sentence are compared to the original sentence, after which the closest version is outputted. A large choice for K will guarantee that semantic meaning is preserved with high probability, but at a large computational cost (each candidate regeneration must be embedded with a non-masked LM for comparison).

An alternative approach is to choose a distance threshold and compare candidate replacements to the original until that distance threshold is reached. Thus, if the second candidate for replacement is very close to the original sentence, no more candidate embeddings will be created.

Formally, for a distance threshold t, a full-sentence embedding f that transforms a sentence of max length L to a matrix of length $L \times D$, a generation G, and a sequence of regeneration candidates G_i that replace the jth token, the chosen replacement is $min(i : ||f(G_i)_j - f(G)_j||_2 < t)$.

It may at first seem that t will be chosen arbitrarily. However, the threshold t can be chosen in a data-driven manner, thanks to the CoinCo dataset. Iterating over all substitution instances of the CoinCo dataset and taking the Euclidean distance between transform embedding of the original and replacement sentences, a

reasonable, arbitrary quantile of the distances can be chosen as the threshold for the replacement procedure. The threshold value is the rounded mean distance between a sentence pre-replacement and post-replacement, which is 2.

4 Results

Results are primarily presented for the case of hard watermarking with gpt-2, as the rate limit of access to the gpt-4 model made it impossible to obtain a large dataset of blacklisted-text generations within a small amount of time.

Given a watermark detection test, a generation, and a dewatermarking procedure, I define a **flip** as the event that the watermark test is rejected for the original generation but not rejected for the dewatermarked version. The flip rate is thus the percentage of generations which are originally watermark-detected and not watermark-detected after regeneration.

Mathematically, if G is a generation from the distribution L of AI generations (possibly constrained to a single language model), $Z_T(G)$ is the Z-statistic of evaluating a watermark test T on G, C is the cutoff for the rejection region, and $D(G)$ is the dewatermarked version of G, then the flip rate F is defined as

$$F(D, T, C, L) = P_{G \sim L}(Z_T(D(G)) > C) \mid Z_T(G) < C)$$

4.1 Masked Bidirectional Language Modeling Ranked by Embedding Similarity

Dewatermarking a Generation with Approach 1

Prompt: It was a dark and stormy night

- Original: It was a dark and stormy night but that didn't stop me from hugging an airline train. By 5 that train was outside slowing as one woman filled me so after catching Singapore Airlines, Flight 1234 to San Francisco
- Dewatermarked: This was a dark and stormy time but that didnna stop travelers from hugging an express train. By 5 that train stood outside slowing as one woman informed me so after catching United Airlines, Route 1134 to San Francisco

Prompt: My mama always said life was like a box of chocolates.

- Original: My mama always said life was like a box of chocolates. Can she learn more?). We're fortunate enough that
- Dewatermarked: Her momama recently complained life feltA giant mountain of chile chips. Can she learn more?). We're fortunate enough that

Evaluation: 50 watermarked completions of length 10 tokens were produced from $gpt-2$ with Watermarking Algorithm 1 and a reduction factor of 2 across 3 prompts. Dewatermarking approach 1 increases the Z-statistic of the watermarking detection hypothesis test by 0.95 on average – recall that the Z-statistic is very negative in the alternative space because it compares the number of blacklist violations to the expected value given the null (half the tokens).

Figure 1 shows that this procedure has a higher flip rate for larger cutoffs, which are the tests that are constrained to a low type-1 error.

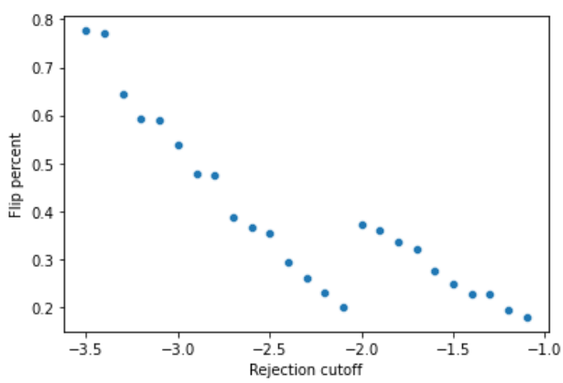

Figure 1: Flip Percentage by Cutoff for Approach 1

4.2 Fine-Tuned Decoding for Word Substitution

Dewatermarking a Generation with Approach 2

Prompt: It was a dark and stormy night

- Original: It was a dark and stormy night but that didn't stop me from hugging an airline train. By 5 that train was outside slowing as one woman filled me so after catching Singapore Airlines, Flight 1234 to San Francisco I
- Dewatermarked: then was a dark and stormydark but that didnjust stopinstance from hugging ancreator train. By 5 that trainmove outside slowing as one womanget me so after catching-marine Airlinesareamove 1234 to San Francisco I

Prompt: My mama always said life was like a box of chocolates.

- Original: My mama always said life was like a box of chocolates. Can she learn more?). We're fortunate enough that
- Dewatermarked: greatgreatamaalsostate lifbassovenbass a box ofbasscriminalates. Can she learn more?). We're fortunate enough that

Once again, 50 watermarked completions of length 10 tokens were produced from gpt-2 with Watermarking Algorithm 1 and a reduction factor of 2 across 3 prompts. Dewatermarking approach 2 increases the Z-statistic by 1.024 on average. Figure 2 shows that the flip rate is larger for larger cutoffs.

Unfortunately, as the exmaple shows, the fine-tuned projection matrix has clearly not converged, as the dewatermarked version is highly nonsensical. A limitation of this approach is that it requires intense training and a very large training set; unfortunately, the CoinCo dataset has only 15,000 target-word instances, which is perhaps not enough to train a projection matrix with over 30 million parameters, even after 100 epochs of training.

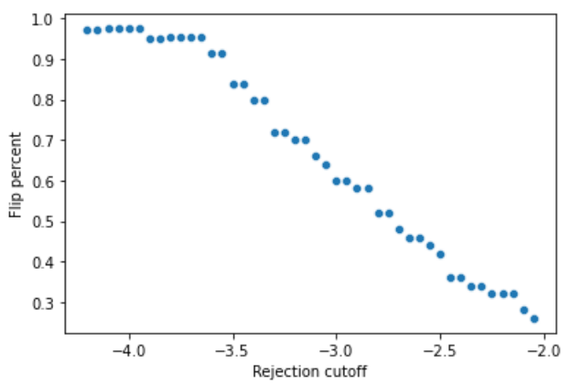

Figure 2: Flip Percentage by Cutoff for Approach 2

4.3 Masked Bidirectional Language Model Constrained to Embedding Distance

Dewatermarking a Generation with Approach 3

Prompt: It was a dark and stormy night

- Original: It was a dark and stormy night but that didn't stop me from hugging an airline train. By 5 that train was outside slowing as one woman filled me so after catching Singapore Airlines, Flight 1234 to San Francisco I
- Dewatermarked: It was a dark and stormy drive but that didn not stop us from hugging an amtrak train. By 5 that bus was outside slowing as one woman told me so after catching United Airlines International Flight 634 to San Francisco I

Prompt: My mama always said life was like a box of chocolates.

- Original: My mama always said life was like a box of chocolates. Can she learn more?). We're fortunate enough that
- Dewatermarked: 'Mama once wrote life's yet another bowl of chile leaves. Can she learn more?). We're fortunate enough that

Once again, 50 watermarked completions of length 10 tokens were produced from $gpt-2$ with Watermarking Algorithm 1 and a reduction factor of 2 across 3 prompts. Dewatermarking approach 3 increases the Z-statistic by 1.024 on average. Figure 3 shows that the flip rate is larger for larger cutoffs.

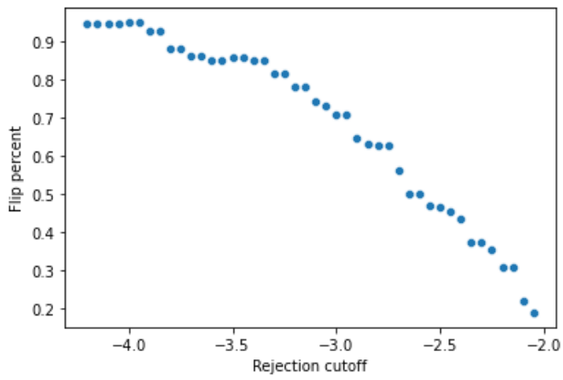

Figure 3: Flip Percentage by Cutoff for Approach 3, Using a Distance Threshold of 2

5 Conclusion

Three approaches were given for removing hard watermarks from AI-generated text from gpt-2: i) masked bidirectional LM ranked by embedding similarity, ii) fine-tuned decoder for replacement, and iii) masked bidirectional LM constrained by embedding distance. All three methods showed strong ability to reduce the power of the watermark detection two-proportion z-test when the test is conducted with a low type-1 error tolerance, and moderate-to-low ability for the more standard, moderate cutoffs.

However, the fine-tuned decoder did not converge to give meaningful outputs. Approaches 1 and 3 anecdotally showed the ability to maintain some semantic similarity, albeit with some grammatical mistakes.

6 Next Steps

A larger dataset of semantic substitutions should be used for fine-tuning the projection matrix, so that Approach 2 may be viable. Instead of tens of thousands of observations, tens of millions would be needed.

A better metric must be defined for evaluating English-to-English semantic similarity, so that the regenerations of Approach 1 and 3 can be systematically compared to each other.

Evaluation of text quality post-dewatermarking was made difficult by the fact that gpt-2's generations are already noticeably low quality, particularly when more tokens are queried. In a sense, the watermark detection problem is useless for gpt-2 because of how conspicuously poor its writing is. The author originally intended to use 3 for generations, but ran out of API access after generating 15 dollars worth of tokens. This study should be replicated for gpt-4 with algorithm 3 by researchers who have more money. The appendix provides examples of watermarked text from gpt-4, which is clearly of higher quality.

References

[1] Yuxin Wen Jonathan Katz Ian Miers Tom Goldstein John Kirchenbauer, Jonas Geiping. A watermark for large language models. *Proceedings of the 40th International Conference on Machine Learning*, pages 17061–17084.

7 Examples of Watermarked and Non-Watermarked Text from gpt-4

Using OpenAI gpt-4 with a temperature of 1.5 and **no blacklisting**.

Prompt: It was a dark and stormy night

- It was a dark and stormy night; the rain fell in torrent except at occasional intervals when it was interrupted by vivid flashes of light **(10 violations)**
- It was a dark and stormy night in the small village of Germany . Tr ember ees quaked , the light from the lights covering the **(13 violations)**

Using OpenAI gpt-4 with a temperature of 2 and blacklisting with a reduction factor of 2

Prompt: It was a dark and stormy night

- It was a dark and stormy night Suspect found found hers:self Facing a murder scene That she couldn't under stand "Something rough " She **(2 violations)**

- It was a dark and stormy night in Brooklyn . Harry Miller was holed up in his small , dingy apartment , squandering the remaining heat from **(3 violations)**