Watermarking LLM Output: A Prompt Based & Token-Level Approach

Admire Madyira, Miro Babin, Aime Cesaire Mugishawayo

1. Introduction

As language models become widely used for writing and communication, it's increasingly important to distinguish between human and AI-generated text. This is crucial for preventing misuse, attributing authorship, and supporting content authenticity.

Watermarking offers a solution by detecting unknown subtle, detectable signal in the model's output. In this project, we explore two strategies for watermarking outputs from **gpt-3.5-turbo**:

- **Prompt-level**: inserting invisible characters into the prompt.
- **Token-level**: biasing the model toward specific tokens during generation.

These methods aim to create watermarked content that remains fluent and natural, yet agnostically verifiable.

Example A

Original Prompt

```
What are the causes of climate change?
```

Watermarked Prompt *(shown with visible marker for illustration purposes— normally invisible to the naked eye)*

```
What are the causes of<U+200B> climate change?
```

⚠️ U+200B (Zero Width Space) is **invisible in practice** and does not affect how the text looks or behaves.

Sample Response *(excerpt with visible marker)*

```
Climate change is caused by human activities such as burning
fossil fuels, deforestation, and global<U+200B> warming...
```

✅ Hidden watermark persists in output — machine-detectable, user-invisible.

Example B

Example Prompt:
What's your favorite color, and why?

Vanilla Response (GPT-3.5 Turbo, No Watermark):
"I really like blue because it's calming and reminds me of the ocean. It's a color that feels peaceful and balanced."

Token-Level Watermarked Response (GPT-3.5 Turbo with Logit Bias Toward "purple"):
"I would say my favorite color is purple. It has a rich, vibrant feel and often symbolizes creativity and depth. There's something unique and elegant about purple that really resonates with me."

2. Methodology

Approach A: Prompt-Based Watermarking

- **Prompt Wrapping**
  Developed a wrapper that injects subtle rules/lexical constraints (e.g. adding zero-width characters) into input prompts.
- **Text Generation**
  Used GPT-3.5-Turbo to generate completions from both original and watermarked prompts.
- **Watermark Detection**

  Used the BERT classifier to detect watermarks in text
  Future: Evaluate how well the watermark signal survives paraphrasing and edits by measuring classifier detection rates after transformation.

Approach B: Token-Level Watermarking

- **Greenlist & Blacklist Setup**
  A set of "greenlisted" tokens is are assigned **positive logit biases,** making them more likely to appear. A blacklist is assigned **negative biases**, making them less likely to appear.
- **Output Generation**
  Produced ~2,000 watermarked completions using the OpenAI Chat API (gpt-3.5-turbo) with controlled sampling.
- **Watermark Detection**

  Used the BERT classifier to detect watermarks in text

Evaluation

- **Dataset**
  Curated ~2,000 short prompts from a HuggingFace dataset for controlled generation across conditions.
- **Automated Detection**
  Trained a lightweight BERTa classifier on non-watermarked output, token-level watermarked output and prompt-based outputs and used to detect between watermarked and non-watermarked text
- **Comparison Metrics**
  - Detection Accuracy
  - Text Quality by sampling output for human inspection

3. Results

We evaluated two watermark detection pipelines based on the nature of the watermarking strategy: **prompt-level watermarking** (inserting zero-width characters) and **token-level watermarking** (biasing sampling logits). For each, we developed both a bag-of-ngrams TF-IDF + logistic regression baseline and a fine-tuned BERT-based classifier. Models were trained to distinguish between watermarked and non-watermarked outputs from a large language model (LLM), given only the generated response as input.

Prompt-Level Watermark Detection

A byte-pair and character-level TF-IDF representations combined with logistic regression achieved test accuracies of **65%**. Fine-tuning a BERT classifier further improved performance to **85%** accuracy, demonstrating that deep models can pick up on latent patterns left by prompt-level perturbations.

Token-Level Watermark Detection

As expected, the watermark signal is more diffuse. The TF-IDF baseline model struggled to generalize beyond surface-level token statistics, achieving only **30%** test accuracy. However, the fine-tuned BERT classifier performed substantially better, achieving **65%** test accuracy. These results suggest that distributional cues in the text are detectable with the right model architecture, even in the absence of explicit token markers.

Watermarking Type	Model	Train Accuracy	Test Accuracy
Prompt-Level (ZWC)	TF-IDF + Logistic Regression	90%	65%
	BERT (fine-tuned)	95%	85%
Token-Level (Logit Bias)	TF-IDF + Logistic Regression	90%	29%
	BERT (fine-tuned)	64%	65%

Acknowledgements: Professor Shira Wein