Advancing Beyond Identification: Multi-bit Watermark for Large Language Models

> https://arxiv.org/abs/2308.00221

- 动机：现有的水印方法都是旨在分辨出某段文本是human-text还是model generated。但是在一些比较严重的场景，如散播虚假新闻等场景下，我们不仅需要分辨出是否是人写的，如果发现是model generated，还希望能更多地追溯到操作者的信息，这样能够更好地惩罚造谣者。在LLMs中植入multi-bit watermark，能够蕴含更多的信息。
- 方法：在zero-bit watermark（只划分red/green list）的基础上，把red/green list扩展成COLOR list。对每一个窗口，还是用伪随机hash的方法，划分下一个position的各个颜色的list。
  - 加入水印的流程
    - 对于一个$\tilde{b}$ bit的信息，先把它chunk成$b = \frac{\tilde{b}}{\lfloor log_2 r \rfloor}$块，每一块对应0到$r-1$中的一个数
    - 在generate下一个token的时候，先从这b块中sample出一块作为此次要加入的信息（即$p \leftarrow sample([0, b-1]), M_r[p] \in [0, r-1]$）
    - 根据前一个窗口的token，用伪随机hash出下一个位置的COLOR list
    - 根据第二点中sample出的信息（0到r-1中的一个数），给对应的color list中的词加bias
  - 从文本中检测M_r

    M_r在每个position的预测当中保持一致，其实就相当于一个API key。每个使用者有一个不同的M_r，这样如果出现了造谣的情况，用Message Extraction算法就能从生成的文本中提取出这个unique的M_r，从而对应到使用者，可以执行惩处或者封号等操作。

    检测算法：

    思想很simple，先把每个位置的颜色算出来，统计每种颜色数量；然后用最大概率反推M_r中每一位的信息。

Algorithm 1: Watermark Extraction

Input: Watermarked text $X_{1:T}$
Output: Predicted message $\hat{M}$, number of colorlisted tokens c

```
/* Initialize counter for every position */
for p in [0, b'] do
    for m in [0, r-1] do
        COUNT_p[m] = 0
/* Count whether token is in colored lists */
for t in [h+1, T] do
    s = f(X_{t-h:t-1})
    p ← sample([0, r-1])
    for m in [0, r-1] do
        Permute V_t using m as seed
        if X_t ∈ G_t^m then
            COUNT_p[m] += 1
/* Predict message */
M̂_r = " "
c = 0
for p in [0, b'] do
    m̂ ← argmax(COUNT_p)
    M̂_r += str(m̂)
    c += max(COUNT_p)
Get bit message M̂ by converting M̂_r
return M̂, c
```

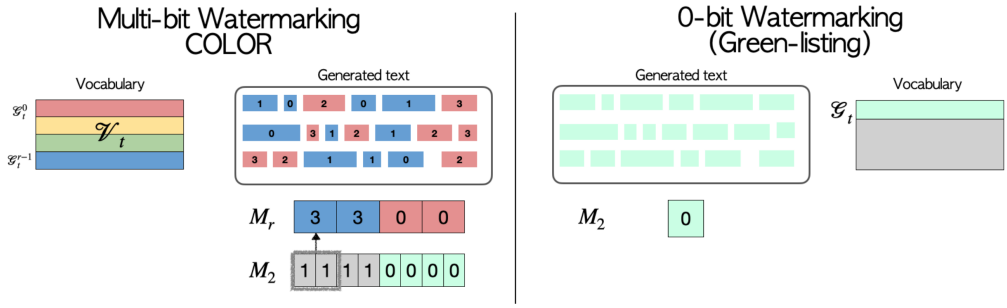
- 分析
  - 当$r = 2, \tilde{b} = 1$时，multi-bit就退化成red/green list
- issue：
  - issue 1

个人感觉把red/green list叫做zero-bit不合适，应该叫做single-bit，因为其实整个message是一个1 bit的0信息，每个位置都对green list加bias

- issue 2

  该图对应的参数应为：$r = 4, \tilde{b} = 8, b = \frac{\tilde{b}}{\lfloor log_2 r \rfloor} = 4$

  文章中写的有误。 when $r = 4$ and $b = 8$,



- issue 3

1. Compute hash of tokens $s = f(X_{t-h:t-1})$. Use s to seed a random number generator.
2. $p \leftarrow \texttt{sample}([0, r-1])$
3. $\texttt{m} \leftarrow M_r[p]$
4. Permute vocabulary $\mathcal{V}_t$ using s as seed.
5. Partition $\mathcal{V}_t = [\mathcal{G}_t^0, \cdots, \mathcal{G}_t^{r-1}]$ discarding remainders if any.
6. Add δ to token logits in $\mathcal{G}_t^{\texttt{m}}$.

第二步应该是$p \leftarrow sample([0, b-1])$