A Watermark for Large Language Models

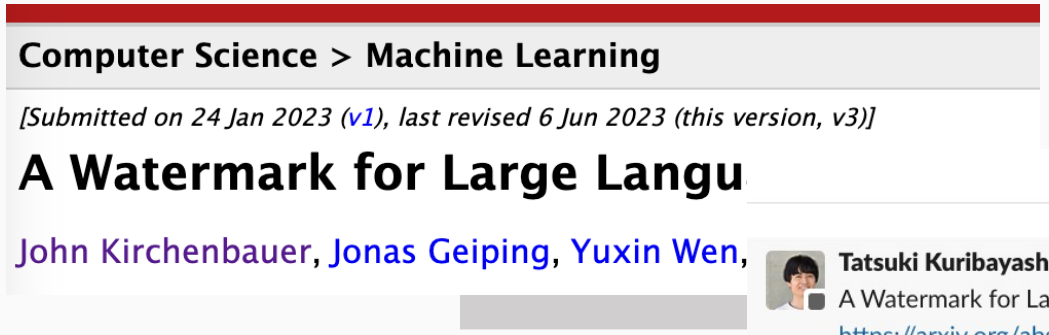
John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz,
Ian Miers, Tom Goldstein (ICML2023)

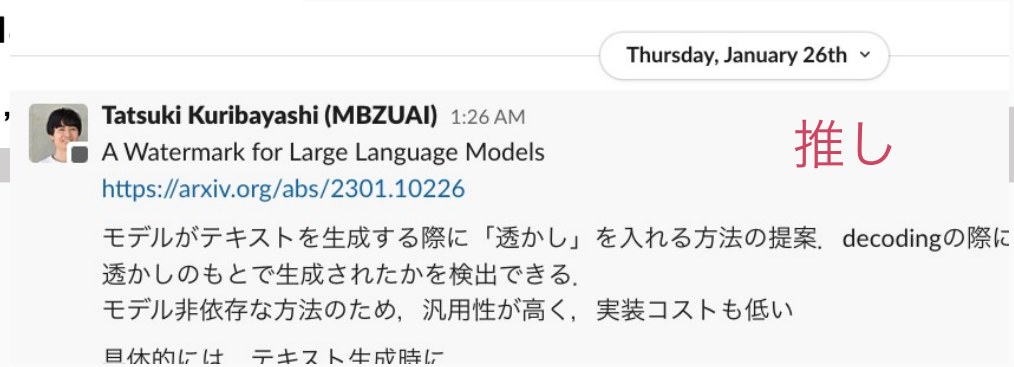
紹介者： 栗林樹生 (MBZUAI)

ある文章が言語モデルによって生成されたものか検知できる世界なら嬉しい

なぜ？

- 自動生成されたテキストの悪用（例：フェイクニュース作成）検知
  - 「あなたのところの言語モデルが迷惑をかけています」という難癖に対する防衛
- 新たな言語モデルを訓練する際の訓練データフィルタリング
- 文章の信頼性（人間によって書かれたか）を判断する一つの指標





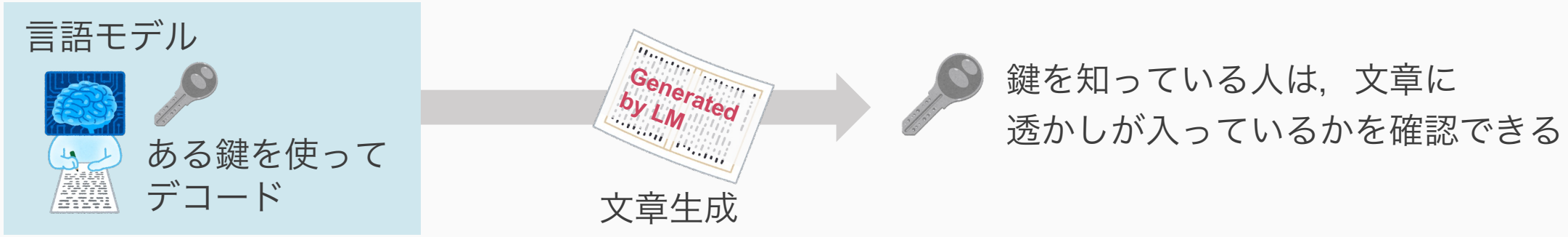
推し

2023/07 ICML
outstanding paper

感慨

やること・やらないこと

【やること】
言語モデルからテキストを生成する際に「透かし」を入れる
---proactive-imprinting

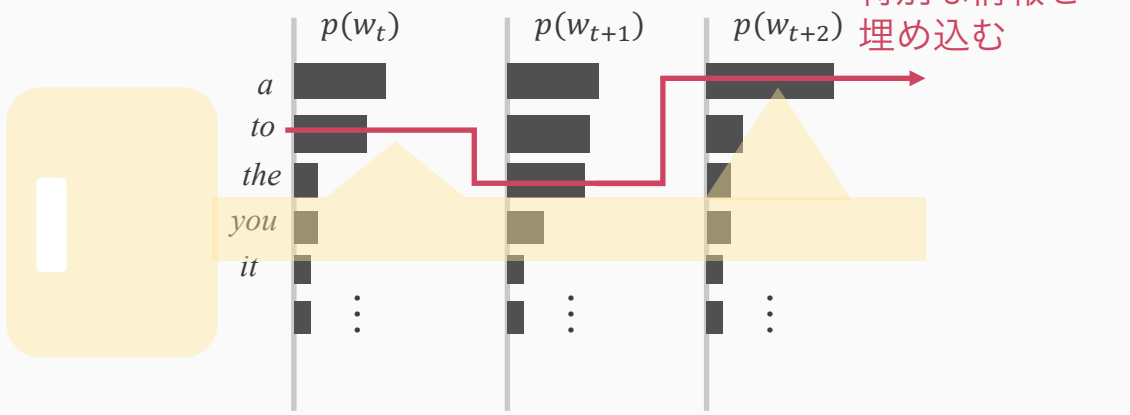


【やらないこと】
- 言語モデル/人間のどちらによって書かれたか文章化を分類するモデル作成（GPTZeroなど）
  ---post-hoc detection
- 任意のテキストに後から透かしを入れる （古くからのやり方，今回はフォーカス外）
  ---post-hoc imprinting
- API提供者がモデルの生成履歴を保存し，照合して検知する （プライバシー問題あり）
  ---retrieval-based detection

提案法の方針

言語モデルから単語をサンプリングする際に特殊なバイアスをかける

提案アイディアの雑なイメージ



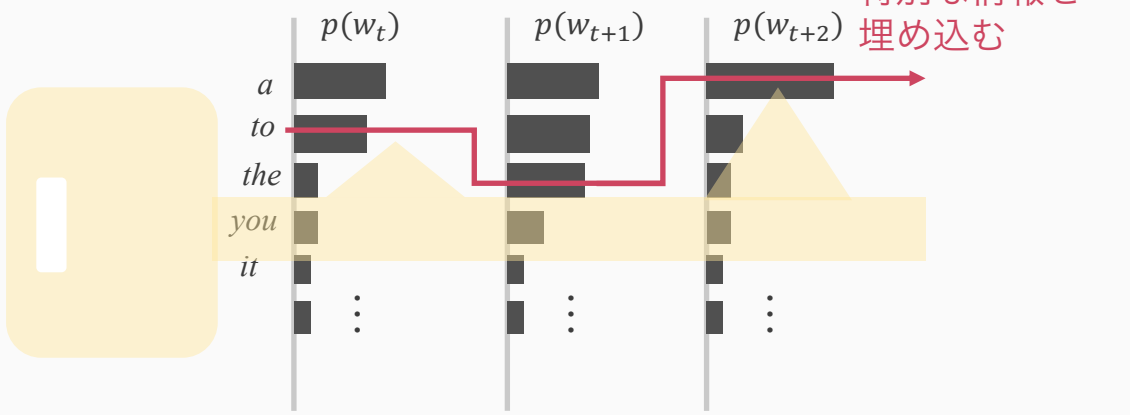
提案法の方針

言語モデルから単語をサンプリングする際に特殊なバイアスをかける

以下をなるべく満たしたい：

- 文章の質を悪化させない
- 透かしを消しづらい
- 透かしの検出が低コスト
  - cf. 検出に大規模言語モデルが必要
- 検出力が高い（第二種の過誤が生じにくい；モデルの生成物を検知しこぼさない）
  - 少ないトークン数で検出できる

提案アイディアの雑なイメージ

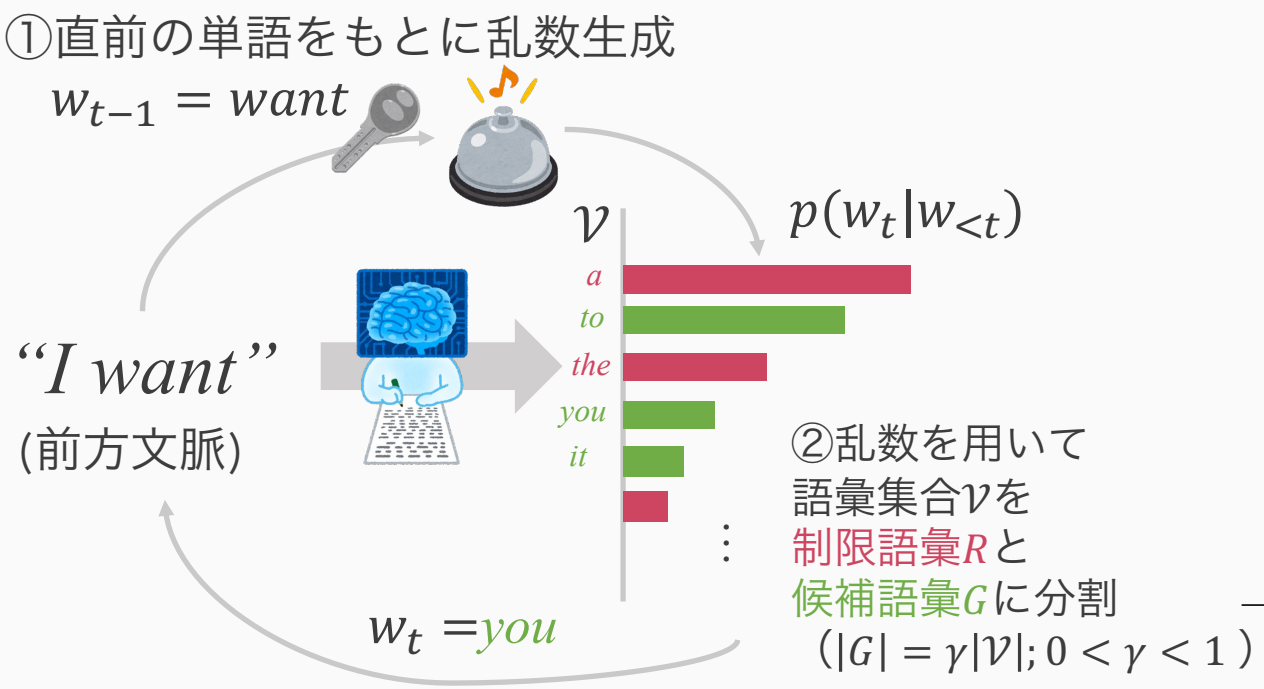


良くない例：なるべく文字数が多い単語が使われるように単語をサンプリングする（透かしを入れる）．
単語の平均長を測ることで透かしが入っているかを判断する．

👎生成された文章はおそらく極めて読みづらい．
👎長い単語を使う方針が推測できれば，短い単語に置き換えていくことで透かしを消せる．

提案法（ハード制約版）

【透かしの入れ方】
各単語を生成する際の候補語彙Gを
直前単語・ハッシュ関数・乱数生成器から定める



Algorithm 1 Text Generation with Hard Red List

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

1. Apply the language model to prior tokens $s^{(-N_p)} \cdots s^{(t-1)}$ 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

※続編[Kirchenbauer+23]では，直前単語以外から乱数を決めることも検討．概ね上述のシンプルな方法で良い

提案法（ハード制約版）

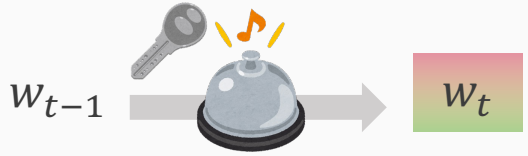
【透かしの入れ方】
各単語を生成する際の候補語彙Gを
直前単語・ハッシュ関数・乱数生成器から定める

①直前の単語をもとに乱数生成

②乱数を用いて
語彙集合$\mathcal{V}$を
制限語彙Rと
候補語彙Gに分
($|G| = \gamma|\mathcal{V}|; 0$

③候補語彙Gから単語をサンプル

【検知方法】
各(w_{t-1}, w_t)について，w_tがGに含まれるかを確認.
Gに含まれる事象が有意に頻発しているか二項検定（$\pi_0 = \gamma$）



ハッシュ関数とランダム
集合分割器があればよい
（👍 元の言語モデルは不要）

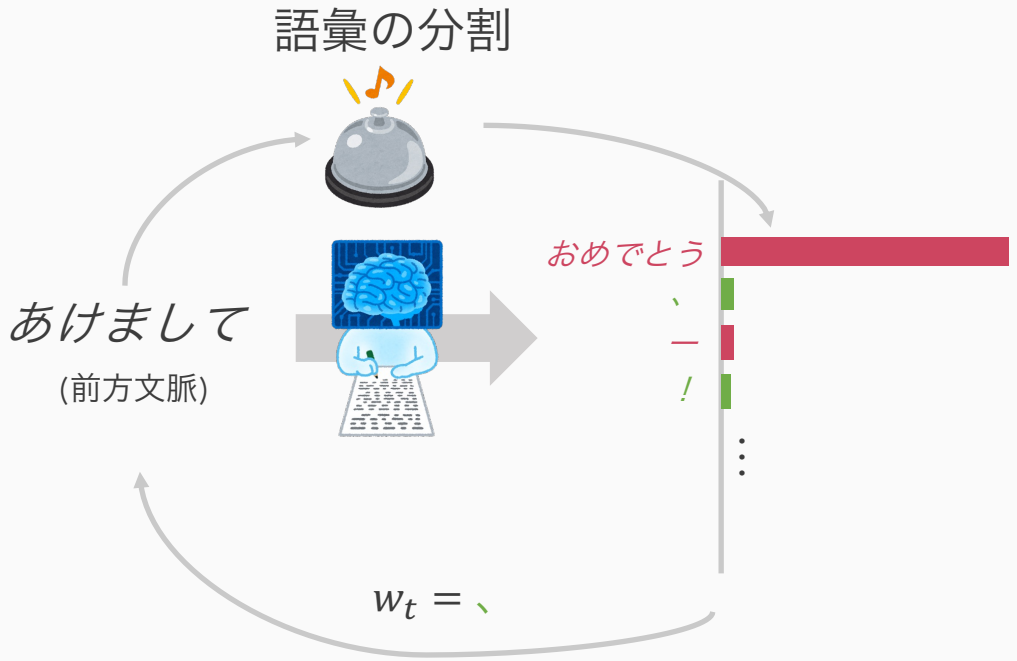
Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

※ハード版の場合，
緑一色になる

※最低でも全体の
1/4トークンを
書き換えないと
緑色トークンを1/2
まで減らせない

限界

- エントロピーの低い部分に透かしを入れると文章の質が悪化する



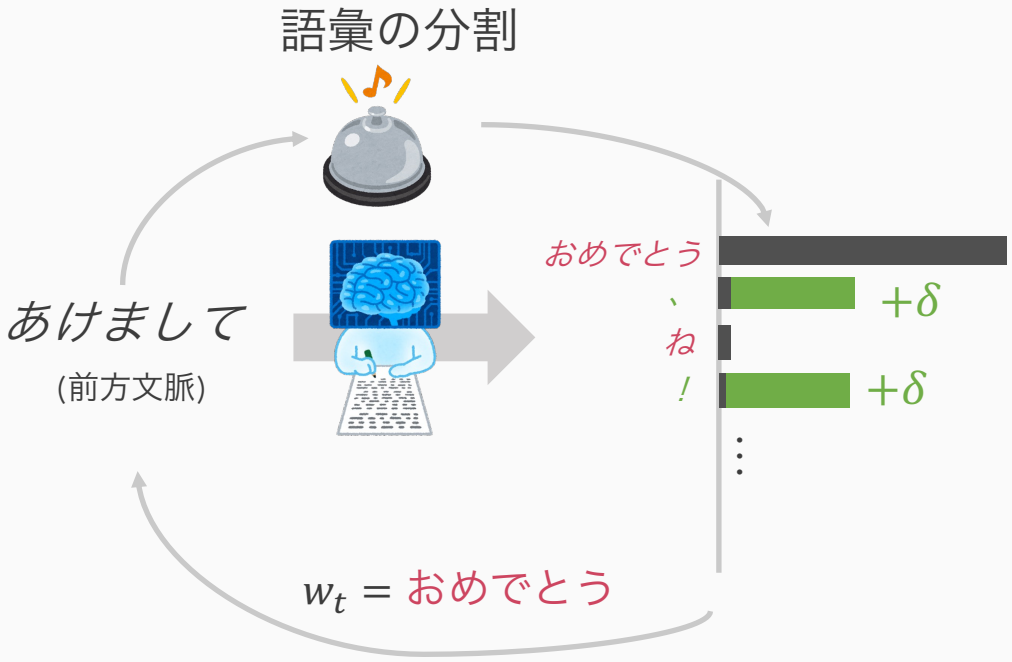
「これしかない」という次単語候補が
偶然制限語彙に入り、文章の質が落ちる
可能性がある

特にプログラム補完などの場面では、
次単語が一意に定まることは起き得る

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

提案法（ソフト制約版）

- 制限語彙を排除するのではなく，候補語彙のロジットに下駄δを履かせる．
  エントロピーの高い箇所で透かしが入り，文章の質の悪化を防げそう．



エントロピーの高い文章ほど，たくさん透かしを入れられる（第一種の過誤が減る）

単語確率分布のスパイクエントロピーsというものを導入し（気持ちはシャノンエントロピーに近い），
長さTの文章に入る推奨語彙の数$|s|_G$の期待値の下界を説明

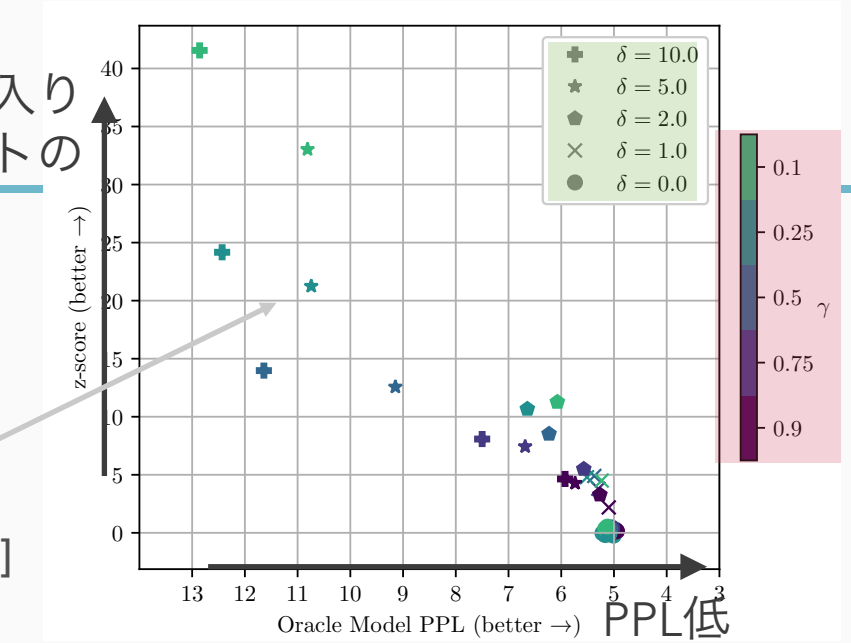
$$\mathbb{E}|s|_G \geq \frac{\gamma T e^{\delta}}{1+(e^{\delta}-1)\gamma} S$$

※続編[Kirchenbauer+23]では，直前単語以外から乱数を決めることも検討．概ね上述のシンプルな方法で良い

実験・分析（抜粋）

【性質の確認（驚きはない）】

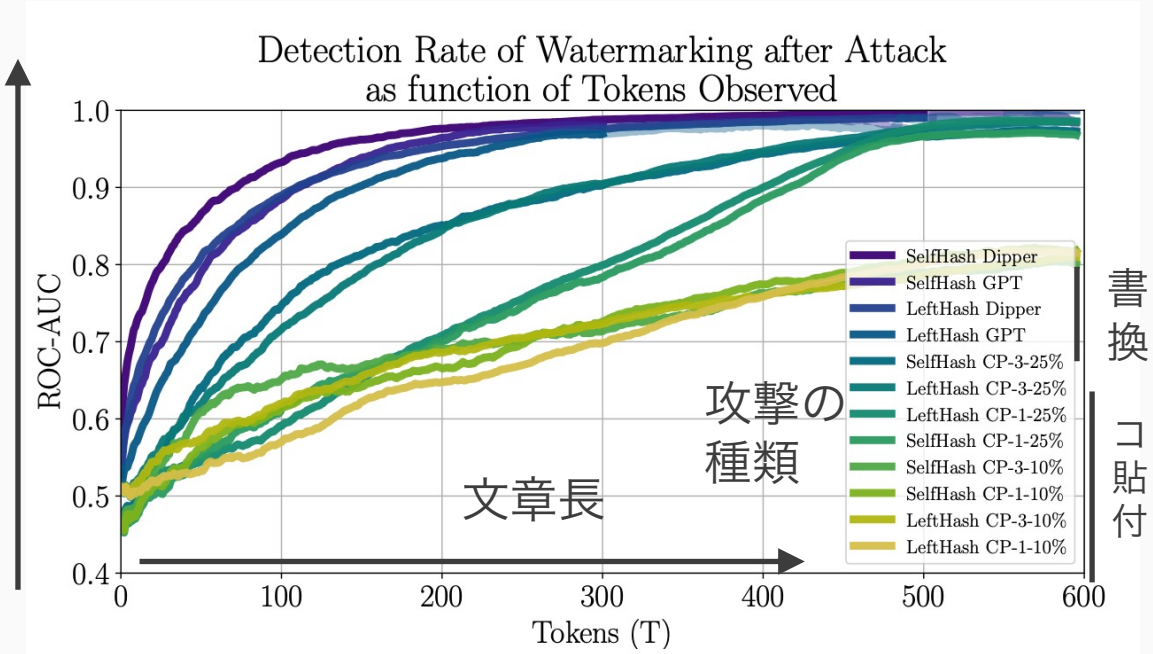
- 透かしの「強さ」（制限語彙割合γ，候補語彙の下駄δ）とPPLはトレードオフ
  - 透かしを強く入れても，人間の評価（透かしあり・なし どっちのテキストを好むか）に大きな影響は与えない [続編Appendix 9.1]



- 文章が長いほど透かし入り文章のz-score高
  - 128tokenもあれば経験上98.4%の透かしテキストは検出できた（OPT）

【頑健性】

- 透かし入り文章をGPT3.5や人間で書き換えただけでは，透かしが完全に消えない [続編]
  - 透かし入りテキストをコピーして透かしなしテキストの一部に混ぜ込まれる状況が対処しにくい



感想

- A Watermark **for Large Language Models** と言っているが,
  仕組み上は任意の単方向デコーダに適応可能，実装簡単
- 社会実装する場合，運用方法について更に議論が必要そう
  - 透かし入りモデルAPIと同時に透かし検出器もAPIで公開するという運用が安牌？
    - 鍵がバレずに，一般ユーザも透かしを検知可能
  - モデルをパラメータごと公開する場合は、ユーザに透かしを入れて貰う必要がある
    - ユーザがわざわざ透かしを入れるインセンティブは？
    - 各々の鍵（ハッシュ関数・乱数生成器）が乱立しすると，検定を繰り返す必要が出てしまう（誰が管理？）
- 続編： On the Reliability of Watermarks for Large Language Models
  (Kirchenbauer+,23; arXiv) もチェック
  - 提案手法の亜種，検定方法の改善，他手法との比較など