Ten Words Only Still Help: Improving Black-Box AI-Generated Text Detection via Proxy-Guided Efficient Re-Sampling

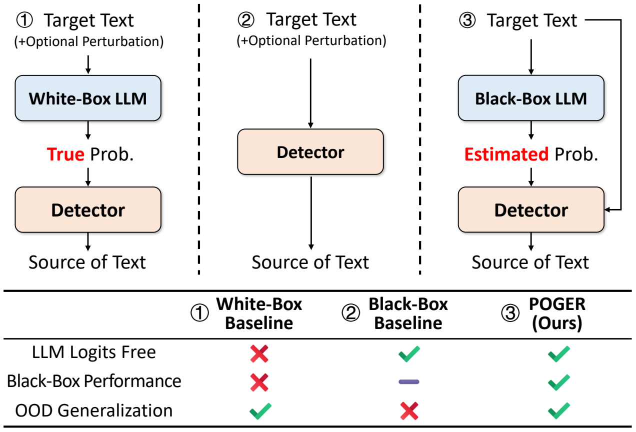
Yuhui Shi[1,2] , **Qiang Sheng**[1] , **Juan Cao**[1,2] , **Hao Mi**[1,2] , **Beizhe Hu**[1,2] , **Danding Wang**[1]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences,
Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{shiyuhui22s, shengqiang18z, caojuan, hubeizhe21s, wangdanding}@ict.ac.cn, mihao1018@gmail.com

Abstract

With the rapidly increasing application of large language models (LLMs), their abuse has caused many undesirable societal problems such as fake news, academic dishonesty, and information pollution. This makes AI-generated text (AIGT) detection of great importance. Among existing methods, white-box methods are generally superior to black-box methods in terms of performance and generalizability, but they require access to LLMs' internal states and are not applicable to black-box settings. In this paper, we propose to estimate word generation probabilities as pseudo white-box features via multiple re-sampling to help improve AIGT detection under the black-box setting. Specifically, we design **POGER**, a proxy-guided efficient re-sampling method, which selects a small subset of representative words (e.g., 10 words) for performing multiple re-sampling in black-box AIGT detection. Experiments on datasets containing texts from humans and seven LLMs show that POGER outperforms all baselines in macro F1 under black-box, partial white-box, and out-of-distribution settings and maintains lower re-sampling costs than its existing counterparts.

1 Introduction

Recent breakthroughs in large language models (LLMs) have significantly improved the quality of AI-generated text (AIGT) and further boosted applications in diverse scenarios. People can easily instruct LLM-supported services like ChatGPT [OpenAI, 2022] to generate texts that are almost imperceptible to humans [Jakesch *et al.*, 2023; Uchendu *et al.*, 2023]. Though LLMs bring much convenience, new societal threats caused by their abuses also emerged: Political manipulators produce AI-generated fake news to risk democracy [Lucas *et al.*, 2023]; students cheat by submitting AI-generated works without paying expected efforts [Bohacek, 2023]; and content farms accelerate information pollution with AI-generated low-quality articles [Brewster *et al.*, 2023]. To build the first barrier against such threats, developing techniques for detecting AI-generated text is of urgent need.



	① White-Box Baseline	② Black-Box Baseline	③ POGER (Ours)
LLM Logits Free	✗	✓	✓
Black-Box Performance	✗	–	✓
OOD Generalization	✓	✗	✓

Figure 1: Paradigm comparison between our proposed POGER and existing white-box/black-box methods. POGER does not require LLMs' internal states like output logits and performs better than the other types of baselines under black-box and out-of-distribution (OOD) settings.

AI-generated text detection is generally defined as a binary or multiclass classification task. The former is to distinguish human-written and AI-generated text, while the latter subsequently recognizes which LLM generates the given text (usually for forensics needs). According to whether the detector can access the source LLM's internal states [Yang *et al.*, 2023b], existing methods can be categorized as white-box and black-box methods. White-box methods [Mitchell *et al.*, 2023; Li *et al.*, 2023a] distinguish LLMs using delicate features reflected by internal states like output token probabilities and usually achieve high detection performance, but its applicability is limited due to the widely existing unavailability of internal states in commercial LLM services. In contrast, black-box methods [Guo *et al.*, 2023] require output texts only for feature acquirement and in principle could be applied to any LLM. However, they generally underperform white-box methods and are more likely to suffer generalization issues on texts from a new domain [Bhattacharjee *et al.*, 2023]. Such a dilemma poses a key challenge for effectively detecting AI-generated text in reality.

To address this issue, a possible solution is to estimate features that proved effective in white-box detection for black-box scenarios (see Figure 1). Inspired by the recent study on inferring the decoding strategy of an LLM with the multiple re-sampling [Ippolito *et al.*, 2023], we suppose that the statistics on re-sampling results on black-box LLMs could serve as a good estimation of word output probabilities, which reflects the nuance of internal states among different LLMs and subsequently help improve black-box detection. In this paper, we conduct the first empirical exploration along this line. Preliminarily, we implement a naive but costly solution that prompts the LLM with a continuation instruction multiple times on each position of the given text (*i.e.*, full-text re-sampling). Results show that even using *estimated* word probabilities, the detector still outperforms the typical black-box RoBERTa-based detector by 14.3%, validating the feasibility of re-sampling-based black-box detection.

To reduce the sampling cost and improve the practicality, we further design **POGER**, a proxy-guided efficient re-sampling method for black-box AIGT detection. The core idea of POGER is to select a subset of words possibly indicative of the LLMs' unique word use characteristics from the given text. As LLMs are usually trained on large-scale human language corpora, the word generated with high probability is often similar across different LLMs under the same context, reflecting human language preferences. Instead, words with lower probabilities are more likely to expose LLMs' uniqueness. Therefore, we employ a proxy white-box LLM to nominate words of relatively low probabilities across the text sequence and then preserve the words with low probability estimation error. By performing re-sampling only for the positions of these words, POGER largely reduces the required sampling times while still maintaining the advantages over existing methods in the challenging 8-class black-box setting. Our contributions are as follows:

- We propose to use estimated word generation probabilities to empower black-box AI-generated text detection and empirically show its feasibility.
- We design POGER, a proxy-guided efficient re-sampling method that largely reduces sampling cost and maintains detection performance by recognizing words that reflect LLMs' uniqueness.
- Extensive experiments on texts from humans and seven popular LLMs show the superiority of POGER over existing methods for binary, multiclass, and out-of-distribution detection scenarios.[1]

2 Background

2.1 Task Formulation

Given a text including n words $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, AIGT detection aims to obtain a classifier $f : \boldsymbol{x} \rightarrow y$, where y is the source of $\boldsymbol{x}$. The task can be further categorized into:

1) Binary AIGT Detection: Distinguish whether a text is generated by AI, *i.e.*, $y \in \{\text{human}, \text{AI}\}$.

2) Multiclass AIGT Detection: Distinguish where a text is from human or a specific AI model, *i.e.*, $y \in \{\text{human}, \theta_1, \theta_2, \cdots, \theta_M\}$, where θ_i is an AI model that can generate text. Similar concepts include origin tracing [Li *et al.*, 2023a] and authorship attribution [Uchendu *et al.*, 2020].

2.2 Related Works

The detection of AIGT can be active or passive. Active methods add pre-designed watermarks to LLM-generated text for later identification [Kirchenbauer *et al.*, 2023; Yoo *et al.*, 2023; Liu *et al.*, 2023; Gu *et al.*, 2023; Wang *et al.*, 2023a]. They show promising results but require extra efforts from stakeholders like LLM providers, not applicable to non-watermarked LLMs. We focus on passive detection, which is more flexible as it operates without modifying LLM workflow. They could be categorized as:

White-box detection methods exploit information from probabilities, which reflects the essentials of language modeling. Earlier works use overall probability [Solaiman *et al.*, 2019], perplexity [Beresneva, 2016; Tian, 2023], or entropy [Lavergne *et al.*, 2008] of the given text on LMs as features. Subsequent works focus on finer-grained token-level probabilities [Gehrmann *et al.*, 2019; Verma *et al.*, 2023]. For example, Sniffer [Li *et al.*, 2023a] and SeqXGPT [Wang *et al.*, 2023b] utilize the probability lists of token sequences on each candidate LLM for multiclass detection. To obtain richer probability information, recent works also consider perturbation of given text on candidate LLMs through mask-filling [Mitchell *et al.*, 2023] or re-generating [Yang *et al.*, 2023a]. White-box detection methods generally perform better and more robustly than black-box ones [Wang *et al.*, 2023b], but the required access to LLMs' internal states largely limits their application to black-box LLMs. Some variants use other models as proxies but performances drop significantly [Mitchell *et al.*, 2023].

Black-box detection methods typically mine effective features from the given text based on semantic representation [Guo *et al.*, 2023; Chen *et al.*, 2023; Zhan *et al.*, 2023] or stylistic expert knowledge [Fröhling and Zubiaga, 2021; Aich *et al.*, 2022]. Recent works [Yang *et al.*, 2023a; Yu *et al.*, 2023] compute the distance between the given text and re-generated texts to reflect the familiarity the candidate LLM has with the given text. Black-box methods have better applicability, but their performance and generalizability generally fall behind white-box ones, especially for multiclass tasks [Li *et al.*, 2023b]. Our POGER combines both their advantages of applicability under the black-box setting and effectiveness brought by (estimated) white-box features.

3 Preliminary Study on Re-Sampling-Based Black-Box AIGT Detection

A recent study reveals that multiple re-sampling could be used to infer the internal decoding strategy of LLMs [Ippolito *et al.*, 2023] under black-box access. Inspired by this, we propose a naive black-box solution that estimates word generation probabilities (proved effective for white-box detection) using multiple re-sampling on the given text to preliminarily validate the feasibility.

3.1 A Naive Solution

We implement the straightforward full-text re-sampling as the naive solution, which samples multiple times at each position of the given text to compute word probabilities on the black-box LLM. For a given text $\boldsymbol{x}$, to obtain the word x_i's proba-

[1] Code, Dataset, and Extended version: https://github.com/ICTMCG/POGER

Domain	Source	# Human	# Generated	Avg. Words
QA	Quora	437	3,059	151.32
	Reddit ELI5	383	2,681	162.39
Writing	IELTS Essay	218	1,526	232.28
	BBC News	288	2,016	196.62
Total		1,326	9,282	177.67

Table 1: Statistics of our AIGT detection datasets.

bility $\hat{p}(x_i|x_{<i})$, we instruct the black-box LLM for N times using the following prompt:

> *Please continue writing the following text, starting from the next word:* $\{x_{<i}\}$

For each prompting, we obtain the generated word at position i by restricting the maximum output length. The estimated probability of x_i given $\{x_{<i}\}$ is computed as the frequency of x_i in the output word set $\{o_j\}_{j=1}^{N}$:

$$\hat{p}(x_i|x_{<i}) = \frac{1}{N}\sum_{j=1}^{N}\mathbb{I}(o_j = x_i), \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function. By repeating the above process for each word in $\boldsymbol{x}$, we obtain an estimated probability list of $\boldsymbol{x}$ on this black-box LLM, denoted as $\hat{\boldsymbol{p}} = \{\hat{p}(x_1), \hat{p}(x_2|x_1), \cdots, \hat{p}(x_n|x_{<n})\}$. With $\hat{\boldsymbol{p}}$ as an alternative input, we can now use white-box methods to validate the feasibility of re-sampling-based detection.

3.2 Experimental Settings

Dataset. Our experiments are based on a dataset consisting of 10,608 text items from humans and seven popular open-sourced or API-based LLMs in two scenarios, covering real-world threats such as low-quality content production, news faking, and student cheating. Table 1 details the statistics.

We first obtain human-written samples from Quora and ELI5 dataset [Fan *et al.*, 2019] for the QA domain and IELTS essay and BBC news dataset for the writing domain [Greene and Cunningham, 2006], respectively (500 each source). Subsequently, we prompt the seven LLMs with questions or writing instructions from the original datasets, including GPT-2 XL [Radford *et al.*, 2019], GPT-J [Wang and Komatsuzaki, 2021], LLaMA-2 13B [Touvron *et al.*, 2023], Alpaca 7B [Taori *et al.*, 2023], Vicuna 13B [Zheng *et al.*, 2023], GPT-3.5 Turbo [OpenAI, 2022], and GPT-4 Turbo [OpenAI, 2023] and collects their responses. We group human and AI texts with the same prompts and remove the groups that contain answers expressing rejection or exceeding 350 words. All samples are split into the train/validation/test sets with a 7:2:1 ratio at the group level.

Metrics. We compute F1 for each class and Macro F1 (MacF1) to evaluate overall performance.

3.3 Results & Analysis

We implement our naive solution combined with two powerful white-box detectors, Sniffer [Li *et al.*, 2023a] and SeqXGPT [Wang *et al.*, 2023b]. To facilitate comparison between white-box and black-box settings, we only consider the five open-sourced LLMs to easily obtain the true probabilities and set a 6-class task in this part.

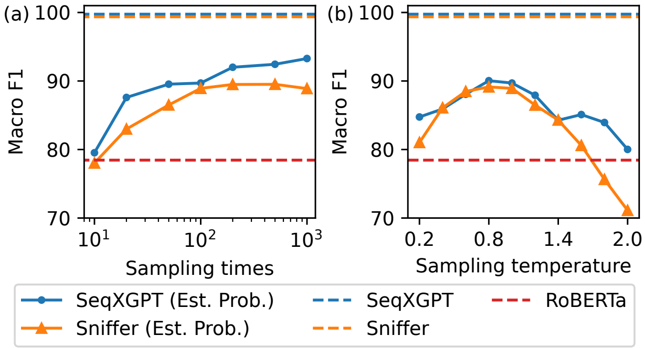


Figure 2: Detection performance using estimated probabilities under different (a) sampling times and (b) sampling temperatures.

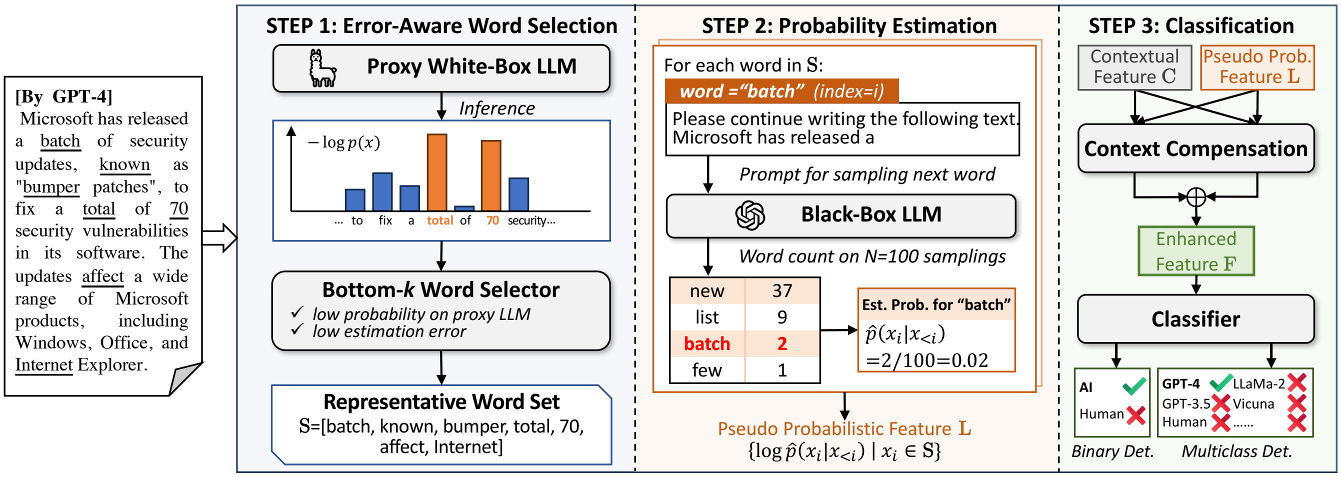
Does Re-Sampling Work? As presented in Figure 2(a), detection performance increases as the sampling times increase for both Sniffer and SeqXGPT. At sampling times of 100, the macro F1 exceeds that of black-box baseline RoBERTa by 14.3%. This result demonstrates the feasibility of estimating word probabilities for black-box AIGT detection, though the requirement of sampling times makes this solution costly.

How does Estimation Error Impact Detection Performance? Inevitably, estimation errors exist with re-sampling of limited times. Though having promising results in Figure 2(a), the detectors still underperform those using true probabilities. To analyze the impact of estimated errors, we adjust the sampling temperature, which changes the probability differences between words and indirectly influences the error. In the resulting Figure 2(b), errors lead to low F1 scores on both left and right sides. On the left, a lower temperature indicates the situation that the target word might not be sampled (estimation probability is 0), resulting in a performance drop. On the right, a higher temperature makes the probabilities of all tokens closer. Even slight sampling randomness causes the ranking of probabilities to change and finally distort the unique characteristics of the LLM. This reveals the importance of error control in probability estimation.

Through the preliminary study, we validated the feasibility of estimated probabilities obtained by re-sampling for black-box AIGT detection. We also identify two issues of the naive solution: 1) the sampling cost on the full text is extremely high; 2) the influence of estimation error is not well controlled. Our improved method to be introduced POGER will tackle these issues.

4 POGER: Proxy-Guided Efficient Re-Sampling for Black-Box AIGT Detection

To tackle the cost and error control issues exposed in Section 3 and design a more practical black-box detector, we propose POGER. Figure 3 presents the overall architecture of POGER. It operates with three steps: First, POGER forms a small word subset from the given text by selecting words of low probabilities and low estimation errors. Subsequently, multiple re-sampling is applied to words in the subset only to obtain a pseudo probabilistic feature. Finally, the feature is



[By GPT-4]
Microsoft has released a batch of security updates, known as "bumper patches", to fix a total of 70 security vulnerabilities in its software. The updates affect a wide range of Microsoft products, including Windows, Office, and Internet Explorer.

new	37
list	9
batch	2
few	1

STEP 1: Error-Aware Word Selection
Proxy White-Box LLM
Inference
$-\log p(x)$
... to fix a total of 70 security...
Bottom-k Word Selector
✓ low probability on proxy LLM
✓ low estimation error
Representative Word Set
S=[batch, known, bumper, total, 70, affect, Internet]
STEP 2: Probability Estimation
For each word in S:
word ="batch" (index=i)
Please continue writing the following text. Microsoft has released a
Prompt for sampling next word
Black-Box LLM
Word count on N=100 samplings
Est. Prob. for "batch"
$\hat{p}(x_i|x_{<i})$ =2/100=0.02
Pseudo Probabilistic Feature $\mathbf{L}$
$\{\log \hat{p}(x_i|x_{<i}) \mid x_i \in \mathrm{S}\}$
STEP 3: Classification
Contextual Feature $\mathbf{C}$
Pseudo Prob. Feature $\mathbf{L}$
Context Compensation
Enhanced Feature $\mathbf{F}$
Classifier
AI ✓
Human ✗
GPT-4 ✓
GPT-3.5 ✗
Human ✗
LLaMa-2 ✗
Vicuna ✗
...... ✗
Binary Det.
Multiclass Det.

Figure 3: Architecture of **POGER**. Given a text piece, POGER operates with three steps: **1) Error-aware word selection**, where a white-box LLM as a proxy to nominate candidate low-probability words and the bottom-k word selector preserves the lowest k word the satisfied estimation error bound; **2) Probability estimation**, where multiple re-sampling is applied to candidate black-box LLMs for the selected k word and a pseudo probabilistic feature $\mathbf{L}$ consisting of estimated probabilities is computed; **3) Classification**, where contextual feature $\mathbf{C}$ is introduced to compensate the context loss in $\mathbf{L}$ to obtain enhanced feature $\mathbf{F}$ for final binary or multiclass AI-generated text detection.

enhanced by compensating contextual information and then fed into a classifier for final detection. Details are as follows.

4.1 Error-Aware Word Selection

Proxy-Based Candidate Nomination. To lower sampling times, we aim to select a small subset of k words from $\boldsymbol{x}$ that reflect the LLM's unique word use characteristics. Here, we use an easy-to-use LM (*e.g.*, GPT-2) as the proxy for candidate nomination. The intuition is as follows: As LLMs are usually trained on large-scale human language corpora, they would learn well on common word use and even different LLMs may output similar texts with high probabilities; in contrast, other words with a lower probability in a text are more likely to expose the unique word use shaped by nuances of LLM training process. Specifically, we use a proxy LM θ to infer on the given text $\boldsymbol{x}$ and obtain token probabilities on it. We transform the list into word-level by computing joint probabilities of corresponding tokens for multi-token words, denoted as $\boldsymbol{p}^\theta = (p_1^\theta, p_2^\theta, \cdots, p_n^\theta)$. A lower p_i^θ would make the x_i more likely to be selected.

Error-aware bottom-k Word Selection. To mitigate the negative impacts of estimated errors on feature effectiveness, we adopt an error-aware bottom-k word selector. For a word x_i with true probability p_i, if the estimated probability of x_i obtained by re-sampling N times is $\hat{p}_i$, the standard error (SE) of $\hat{p}_i$ is given by:

$$\mathrm{SE}(\hat{p}_i) = \sqrt{\frac{p_i(1-p_i)}{N}}. \tag{2}$$

For low-probability words, we constrain a lower bound on their true probability to ensure that the error in the estimated probability does not exceed Δ times itself:

$$\mathrm{SE}(\hat{p}_i) \leq \Delta \cdot p_i \quad \Rightarrow \quad p_i \geq \frac{1}{1+N\Delta^2}. \tag{3}$$

By controlling the relative error, we remove the items in $\boldsymbol{p}^\theta$ that do not meet the error requirements and obtain $\boldsymbol{p}^{\theta\prime}$. Words which are with lowest k probabilities are selected from $\boldsymbol{p}^{\theta\prime}$ using $\mathrm{MINK}(\cdot)$ function and finally form the representative word set S:

$$\boldsymbol{p}^{\theta\prime} = \left\{ p_i \middle| p_i \geq \frac{1}{1+N\Delta^2} \right\}, \tag{4}$$

$$\mathrm{IDX} = \left\{ i \middle| p_i^\theta \in \mathrm{MINK}(\boldsymbol{p}^{\theta\prime}) \right\}, \mathrm{S} = \{x_i | i \in \mathrm{IDX}\}. \tag{5}$$

4.2 Probability Estimation

We again use the sampling and probability calculation process described in Section 3.1, but only for the selected k words in S on the given M candidate black-box LLMs (denoted as $\{\theta_i\}_{i=1}^M$) by N times. For efficiency needs, we constrain the maximum context length as b. We get the pseudo log probabilistic feature matrix $\mathbf{L} = [\boldsymbol{l}_i]_{i=1}^k \in \mathbb{R}^{k\times M}$, where the M-dimentional feature vector for the i-th word is $\boldsymbol{l}_i = \left[\hat{p}_{\theta_j}\left(x_{\mathrm{IDX}[i]}|x_{\mathrm{IDX}[i]-b:\mathrm{IDX}[i]-1}\right)\right]_{j=1}^M$.

4.3 Context-Compensated Classification

In the final step, the source of the target text $\boldsymbol{x}$ is classified based on $\mathbf{L}$. Following [Wang *et al.*, 2023b], we first transform the $\mathbf{L}$ into another $\mathbf{L}' \in \mathbb{R}^{k\times d}$ using convolutional neural network and Transformer to enrich the representation. Furthermore, since we discontinuously select representative words, the information about the local context around the word and their relative positions in the original text is lost. As context compensation, we introduce the contextual semantic representation of the k words to bootstrap the probabilistic representation $\mathbf{L}'$.

Specifically, we input the given text $\boldsymbol{x}$ into RoBERTa [Liu *et al.*, 2019] and obtain the last-layer word representation as $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n\}$. The representations for the k representative words are then mapped to d dimension through a

Method	Human	GPT-2	GPT-J	LLaMA-2	Vicuna	Alpaca	GPT-3.5	GPT-4	MacF1
Partial White-Box Setting									
DNA-GPT White	N/A	62.70	40.79	45.36	30.49	70.18	N/A	N/A	49.91*
Sniffer	96.60	**100.00**	**100.00**	98.49	95.85	**99.23**	75.34	72.65	92.27
SeqXGPT	**98.07**	**100.00**	99.62	**98.88**	**99.62**	98.87	85.93	84.17	95.64
POGER-Mixture	97.32	98.88	99.23	98.11	97.71	98.86	**97.36**	**97.38**	**98.11**
w/o Context Compensation	96.97	99.62	99.23	96.68	94.94	98.48	95.42	95.13	97.06
Black-Box Setting									
RoBERTa	88.24	78.03	86.55	55.47	58.70	59.91	70.63	84.13	72.71
T5-Sentinel	87.29	85.42	88.71	67.78	62.11	69.73	75.79	79.83	77.08
DNA-GPT Black	N/A	38.58	21.56	48.80	33.85	47.15	53.99	39.82	40.53*
Sniffer	87.41	89.82	87.26	29.52	47.62	35.84	34.21	52.63	58.04
SeqXGPT	91.67	89.66	86.77	23.64	46.31	45.64	42.10	62.40	61.02
POGER	**92.49**	**93.75**	**89.96**	**90.49**	**89.30**	**93.82**	**90.98**	**92.59**	**91.67**
w/o Context Compensation	84.21	88.30	80.63	81.88	88.65	91.95	89.49	87.35	86.56

Table 2: F1 scores in two settings for multiclass AIGT detection. The best two scores in each setting are respectively **bolded** and underlined. The shaded area denotes the performance on black-box LLMs. * Because of the nature of DNA-GPT, the macro F1 scores of DNA-GPT White and Black are derived under a pure-white-box setting (5 classes) and a black-box setting without human class (7 classes).

multilayer perceptron (MLP), forming the contextual feature matrix $\mathbf{C} \in \mathbb{R}^{k \times d}$:

$$\mathbf{C} = [\mathrm{MLP}(\mathbf{e}_i)]_{i \in \mathrm{IDX}} \,. \tag{6}$$

Then a bidirectional cross-attention is adopted to build interaction between $\mathbf{C}$ and $\mathbf{L}'$, outputting the enhance feature $\mathbf{F} \in \mathbb{R}^{k \times 2d}$:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \tag{7}$$

$$\mathbf{F} = \mathrm{Attention}\left(\mathbf{L}', \mathbf{C}, \mathbf{C}\right) \oplus \mathrm{Attention}\left(\mathbf{C}, \mathbf{L}', \mathbf{L}'\right), \tag{8}$$

where $\oplus$ is a concatenation operation. Since the representation of each position in $\mathbf{C}$ implies contextual information, this interaction allows relative positional information to be fused into the final enhanced feature.

Finally, $\mathbf{F}$ is fed into another MLP for final classification. The network is optimized using cross-entropy loss.

5 Evaluation

5.1 Experimental Settings

Settings. We continue using the datasets and LLMs introduced in Section 3. To simulate real-world situations, we have two settings: 1) **Partial White-Box Setting** provides true probabilities for the five open-sourced LLMs; and 2) **Black-box Setting** treats all models as black-box LLMs. For the former setting, we provide a variant **POGER-Mixture**, which uses true probabilities from white-box LLMs and estimated ones from black-box LLMs. The estimated probability list is expanded to the same dimension as the true lists by padding with 0. We evaluate for both binary and multiclass detection tasks.

Baselines. **1) GPTZero** [Tian, 2023]: Distinguish between human and generated text using perplexity and burstiness of text; **2) RoBERTa** [Guo *et al.*, 2023]: A widely used and powerful pre-training-based detector; **3) T5-Sentinel** [Chen *et al.*, 2023] Another pretraining-based method for reframing the classification task as a next-token prediction task. **4) DNA-GPT** [Yang *et al.*, 2023a]: Determine the source of text based on multiple re-generation, can works in two forms under black-box and white-box settings. **5) Detect-GPT** [Mitchell *et al.*, 2023]: Determine whether a text is generated by comparing the probability of the original text with a large number of perturbed texts. **6) Sniffer** [Li *et al.*, 2023a]: Determine the source of the text using the contrastive features of the probability lists on each candidate model. **7) SeqXGPT** [Wang *et al.*, 2023b]: Also based on probability lists of the text, but reframes the classification task as a sequence labeling task.

Implementation Details. For POGER and POGER-Mixture, we use GPT-2 Large as the proxy for representative word selection, with maximum error tolerance $\Delta = 1.2$, representative word set size $k = 10$, re-sampling times $N = 100$, and sampling temperature $t = 1.0$. For the five open-source models, we perform re-sampling locally, and for GPT-3.5 Turbo and GPT-4 Turbo, we call OpenAI API for re-sampling and set $max\_tokens = 2$. We input $b = 20$ words before the target word as context. For white-box detection methods under the black-box setting, we employ GPT-Neo 2.7B and LLaMA 7B as proxy probability providers to ensure their proper functionality (some methods require at least two proxies). For zero-shot detection methods, we do a grid search on classification thresholds and report their optimal performance in the search interval.

5.2 Main Results

Multiclass AIGT Detection

Table 2 shows the performance comparison of POGER and its variants with other baselines. Based on the result, we have the following observations:

- POGER and POGER-Mixture outperform all baselines in macro F1 in both the black-box and partial white-box settings. In particular, POGER and POGER-Mixture

Setting	Method	Human	Generated	MacF1
Partial White-Box	DetectGPT	60.00	95.58	77.79
	DNA-GPT White	77.05	97.00	87.03
	SeqXGPT	96.60	99.51	98.06
	POGER-Mixture	**97.69**	**99.67**	**98.68**
Black-Box	RoBERTa	92.06	98.92	95.49
	T5-Sentinel	87.29	98.40	92.85
	DNA-GPT Black	42.08	87.09	64.59
	DetectGPT	44.81	92.89	68.85
	SeqXGPT	92.07	98.86	95.47
	GPTZero	68.42	95.45	81.94
	POGER	**93.89**	**99.14**	**96.51**

Table 3: F1 scores in two settings of binary AIGT detection. The best result under each setting is **bolded**.

outperform all black-box LLMs (shaded in the table) in single-class F1. This demonstrates that POGER has superior performance, especially for black-box detection.

- Compared with the partial white-box setting, all baseline models experience significant performance degradation in the black-box setting. Among the baseline models, semantic-based RoBERTa and T5-Sentinel perform the best, but they still fall behind POGER by more than 15.9% in macro F1, which indicates that POGER could achieve a good balance between applicability and performance.
- We evaluate the effectiveness of the Context Compensation module in POGER and POGER-Mixture in both settings. We can see that, on the one hand, *w/o Context Compensation* brings a decrease of over 5 macro F1 scores in POGER performance, highlighting the significance of this module. On the other hand, even the *w/o Context Compensation* variant of POGER still outperforms all baselines, demonstrating the effectiveness of the resampling strategy we proposed.

Binary AIGT Detection

Table 3 shows the performance of each method in binary detection. Among the baselines, RoBERTa, SeqXGPT, and T5-Sentinel perform well (macro F1 of over 90) while others gain unsatisfying performance. This might be influenced by the method nature of focusing more on distinguishing different LLMs. Still, POGER gains the best performance in both settings, showing its wide applicability in different AIGT detection tasks.

5.3 Out-of-Distribution Results

Due to variations in semantic and stylistic features across different domains in AIGT, existing training-based black-box detection methods often exhibit poor performance in Out-Of-Distribution (OOD) scenarios. We conducted multi-class AIGT detection experiments between two domains within our dataset, where training samples were sourced from one domain and testing was performed on the other domain. The results are presented in Table 4.

Method	In-Dist.	Out-of-Distribution			
		QA→Writing		Writing→QA	
RoBERTa	72.71	54.23	*(-25.42%)*	46.73	*(-35.73%)*
T5-Sentinel	77.08	47.23	*(-38.73%)*	53.19	*(-30.99%)*
Sniffer	58.04	57.50	*(-0.93%)*	53.16	*(-8.41%)*
SeqXGPT	61.02	59.07	*(-3.20%)*	54.94	*(-9.96%)*
POGER	**91.67**	**89.00**	*(-2.91%)*	**84.19**	*(-8.16%)*

Table 4: F1 scores of the OOD experiment. The relative decrease for OOD scenarios over the in-distribution F1 score is shown in the brackets. In-Dist.: In-Distribution.

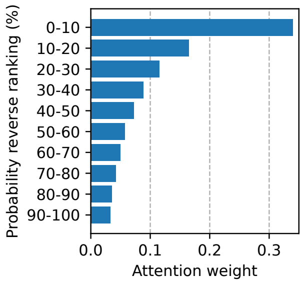


Figure 4: Distribution of attention weight for words in different probability ranking intervals.

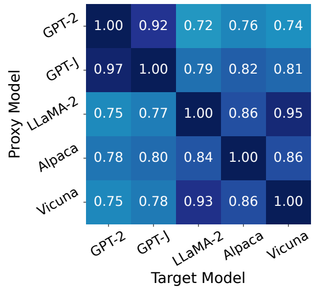


Figure 5: Overlapping Proportion of low-probability words between different LLMs.

It can be seen that POGER still outperforms all baselines, both in terms of generalizing from QA to writing and vice versa. Meanwhile, compared with their respective In-Distribution performance, the relative performance degradation of POGER in the OOD scenario is significantly smaller than the two black-box baselines RoBERTa and T5-Sentinel, and is comparable to the two white-box baselines Sniffer and SeqXGPT. This indicates that, despite being a black-box AIGT detector, POGER benefits from the excellent OOD generalization capabilities inherited from white-box detection methods through the pseudo probabilistic feature.

6 Analysis

6.1 Representativeness of Selected Words

We conduct empirical analysis to validate if our word selection methods will select representative words.

Are Low-Probability Words More Helpful in Detection? We implement a simple white-box AIGT detector with an attention layer that takes true probabilities on candidate LLMs as inputs. Figure 4 shows the attention distribution on words in different probability ranking intervals. We observe that the 10% lowest-probability words gain over 30% of the attention weights. As word probability increases, their attention weights decline, indicating low-probability words play more important roles in AIGT detection.

Are Low-Probability Word Sets Similar Between the Proxy and Candidate LLMs? We use each LLM as a proxy model and the other LLMs as target models and show the proportion of words with the lowest 5% probability on the proxy model whose probability on the target model was in the lowest 20% in Figure 5. Even in the worst case, 72% of low-probability words on the proxy model are also hit in the set of the target model. This suggests that proxy model could be used as a good indicator for word selection.

6.2 Hyperparameter Senstivity

We conduct an analysis of four hyper-parameters on a subset containing text from humans and the five open-sourced LLMs for brevity:

Impact of Maximum Error Tolerance. Figure 6(a) shows the performance impact of the maximum error tolerance Δ in the representative word selector. As Δ increases, we find that the performance curve shows a trend of rising first and then falling. This is due to the fact that when Δ takes a small value, as the selector has a strict constraint on the error of empirical probability, the low-probability words that best reflect the characteristics of the text source are filtered out, rendering the words in set S insufficiently representative. Representative; when Δ takes a large value, the error of empirical probability also increases, resulting in the weakening of the effectiveness of probabilistic features. When Δ is between 1.2 and 2.2, our selector is able to strike a good balance between word representativeness and estimation error.

Impact of Representative Word Set Size & Re-Sampling Times. Intuitively, if the representative word set size k and the re-sampling time N are increased at any cost, the performance of the detector will also increase. However, in practice, we expect POGER to achieve the best possible detection performance while meeting certain cost and efficiency requirements. Therefore, we examine the effect of k and N on POGER performance at the same level of total sampling number on the black-box LLM. Figure 6(b) shows the performance of POGER under different (k, N) pairs when the total sampling number $k \cdot N = 1000$. It is observed that POGER performs best when k is between 10 and 50 (*i.e.*, N is between 20 and 100). When k is too small, the detector obtains too little information about probability lists; and when N is too small, the error-aware selector filters out a large number of words with excessive errors, resulting in the selected words not being representative enough. Both situations result in POGER failing to make accurate classifications.

We find that compared to the version without content compensation, the full POGER has less variation in performance in response to changes in either N, k, or Δ. We believe this is due to the fact that the contextual semantic information is able to compensate for a portion of the performance in cases where the probabilistic features are not effective enough.

Proxy Model for Representative Words Selection. We conduct small-scale experiments using GPT-2 Large, GPT-Neo 2.7B, and LLaMA 7B as proxy models in representative word selection, and the Macro F1s are 85.77, 85.42, and 86.14, respectively, indicating that POGER is not sensitive to the selection of proxy models. It also proves once again that not using the probability value of the proxy model but determining the approximate range of the probability is a proper way of using proxy models for AIGT detection.

6.3 Cost Comparison

With a well-designed re-sampling strategy, we achieve high-performance black-box AIGT detection at a relatively low cost. Although POGER requires re-sampling for each representative word, we only infer for the next position to obtain the generated word instead of repeatedly generating the whole text sequence, thus requiring limited LLM inference cost. If the maximum number of generated tokens is set to 1, the inference length of LLM is even independent of the re-sampling times, since no additional inference beyond prompt is required. Table 5 shows a comparison of the number of inference tokens on a target LLM for POGER and other regeneration-based AIGT detection methods, indicating that POGER's inference cost is similar to DNA-GPT and much less than DetectGPT and Full Sampling.

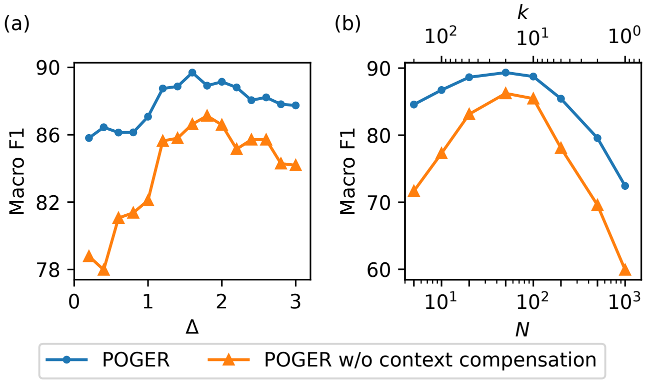


Figure 6: Performance of POGER and its variant under (a) different maximum error tolerances (b) different representative word set sizes & re-sampling times, where the product of N and k is constrained to be equal to 1,000.

Method	# Target LLM Inference Tokens		
	Expression	Typical Value	Ratio
DetectGPT	$n \cdot l$	30,000	× 21.43
DNA-GPT	$r \cdot l + n \cdot (1-r) \cdot l$	1,650	× 1.18
Full Sampling	$l \cdot [l_p + n(m-1)]$	42,000	× 30
POGER	$k \cdot [l_p + n(m-1)]$	1,400	× 1

Table 5: Comparison of the number of inference tokens needed for detection, where n denotes the number of re-generated samples, l denotes the text length, r denotes the truncation ratio in DNA-GPT, k, l_p, and m denote the size of the representative word set, the prompt length, and the maximum number of generated tokens in POGER, respectively. Full Sampling refers to the naive solution in Section 3.1. In the calculation of typical values, l is taken as 300 tokens (about 200 words), and the values of other variables are referred to the original publication.

7 Conclusion and Discussion

In this paper, we proposed to estimate features that proved effective in the white-box setting to help improve black-box AIGT detection. We first developed a naive solution that leverages multiple re-sampling to estimate word generation probabilities for black-box detection. To further reduce the sampling cost, we designed POGER, which leverages a proxy model to select a subset of representative words with an awareness of sampling errors. Experiments on texts from humans and seven LLMs demonstrated the superiority of POGER for binary, multiclass, and OOD scenarios. Further cost analysis indicates that POGER keeps lower re-sampling costs than its counterparts. In the future, we plan to further improve the efficiency by introducing result storage.

Ethical Statement

Considering that white-box and black-box LLMs like LLaMA-2 [Touvron *et al.*, 2023] and GPT-4 [OpenAI, 2023] have been widely used in daily lives, and the AI-generated texts have posed real-world threats and are believed to bring more serious societal harms, our research attempts to propose a new method for AI-generated text detection to help defense against the unknown threats in the future. Considering the performance is not perfect for now, the text that is flagged as AI-generated text by POGER and its counterparts should go through extra checking before making an official accusation regarding rule violations to a certain person in practice.

Acknowledgments

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB068 0202) and the Postdoctoral Fellowship Program of CPSF (GZC20232738). The corresponding author is Qiang Sheng.

References

[Aich *et al.*, 2022] Ankit Aich, Souvik Bhattacharya, and Natalie Parde. Demystifying Neural Fake News via Linguistic Feature-Based Interpretation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6586–6599, 2022.

[Beresneva, 2016] Daria Beresneva. Computer-Generated Text Detection Using Machine Learning: A Systematic Review. In *Proceedings of the 21st International Conference on Applications of Natural Language to Information System*, pages 421–426, 2016.

[Bhattacharjee *et al.*, 2023] Amrita Bhattacharjee, Tharindu Kumarage, Raha Moraffah, and Huan Liu. ConDA: Contrastive Domain Adaptation for AI-generated Text Detection. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 598–610, 2023.

[Bohacek, 2023] Matyas Bohacek. The Unseen A+ Student: Navigating the Impact of Large Language Models in the Classroom. In *ICML 2023 Workshop on Deployment Challenges for Generative AI*, 2023.

[Brewster *et al.*, 2023] Jack Brewster, Zack Fishman, and Elisa Xu. Funding the Next Generation of Content Farms. https://www.newsguardtech.com/misinformation-monitor/june-2023/, 2023. Accessed: 2024-05-15.

[Chen *et al.*, 2023] Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, and Bhiksha Raj. Token Prediction as Implicit Classification to Identify LLM-Generated Text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13112–13120, 2023.

[Fan *et al.*, 2019] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, 2019.

[Fröhling and Zubiaga, 2021] Leon Fröhling and Arkaitz Zubiaga. Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover. *PeerJ Computer Science*, 7:e443, 2021.

[Gehrmann *et al.*, 2019] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, 2019.

[Greene and Cunningham, 2006] Derek Greene and Pádraig Cunningham. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 377–384, 2006.

[Gu *et al.*, 2023] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the Learnability of Watermarks for Language Models. *arXiv preprint arXiv:2312.04469*, 2023.

[Guo *et al.*, 2023] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. *arXiv preprint arXiv:2301.07597*, 2023.

[Ippolito *et al.*, 2023] Daphne Ippolito, Nicholas Carlini, Katherine Lee, Milad Nasr, and Yun William Yu. Reverse-Engineering Decoding Strategies Given Blackbox Access to a Language Generation System. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 396–406, 2023.

[Jakesch *et al.*, 2023] Maurice Jakesch, Jeffrey T Hancock, and Mor Naaman. Human Heuristics for AI-Generated Language are Flawed. *Proceedings of the National Academy of Sciences*, 120(11):e2208839120, 2023.

[Kirchenbauer *et al.*, 2023] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 17061–17084, 2023.

[Lavergne *et al.*, 2008] Thomas Lavergne, Tanguy Urvoy, and François Yvon. Detecting Fake Content with Relative Entropy Scoring. In *Proceedings of the 2008 International Conference on Uncovering Plagiarism, Authorship and Social Software Misuse-Volume 377*, pages 27–31, 2008.

[Li *et al.*, 2023a] Linyang Li, Pengyu Wang, Ke Ren, Tianxiang Sun, and Xipeng Qiu. Origin Tracing and Detecting of LLMs. *arXiv preprint arXiv:2304.14072*, 2023.

[Li *et al.*, 2023b] Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. Deepfake Text Detection in the Wild. *arXiv preprint arXiv:2305.13242*, 2023.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[Liu *et al.*, 2023] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A Semantic Invariant Robust Watermark for Large Language Models. *arXiv preprint arXiv:2310.06356*, 2023.

[Lucas *et al.*, 2023] Jason Lucas, Adaku Uchendu, Michiharu Yamashita, Jooyoung Lee, Shaurya Rohatgi, and Dongwon Lee. Fighting Fire with Fire: The Dual Role of LLMs in Crafting and Detecting Elusive Disinformation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14279–14305, 2023.

[Mitchell *et al.*, 2023] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[OpenAI, 2022] OpenAI. ChatGPT: Optimizing language models for dialogue. https://openai.com/index/chatgpt/, 2022. Accessed: 2024-05-15.

[OpenAI, 2023] OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 2019.

[Solaiman *et al.*, 2019] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

[Taori *et al.*, 2023] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023. Accessed: 2024-05-15.

[Tian, 2023] Edward Tian. GPTZero: An AI Text Detector. https://gptzero.me/, 2023. Accessed: 2024-05-15.

[Touvron *et al.*, 2023] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[Uchendu *et al.*, 2020] Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8384–8395, 2020.

[Uchendu *et al.*, 2023] Adaku Uchendu, Jooyoung Lee, Hua Shen, Thai Le, Ting-Hao 'Kenneth' Huang, and Dongwon Lee. Does Human Collaboration Enhance the Accuracy of Identifying LLM-Generated Deepfake Texts? In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 11, pages 163–174, 2023.

[Verma *et al.*, 2023] Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. Ghostbuster: Detecting Text Ghostwritten by Large Language Models. *arXiv preprint arXiv:2305.15047*, 2023.

[Wang and Komatsuzaki, 2021] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, 2021. Accessed: 2024-05-15.

[Wang *et al.*, 2023a] Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*, 2023.

[Wang *et al.*, 2023b] Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. SeqXGPT: Sentence-Level AI-Generated Text Detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1144–1156, 2023.

[Yang *et al.*, 2023a] Xianjun Yang, Wei Cheng, Linda Petzold, William Yang Wang, and Haifeng Chen. DNA-GPT: Divergent N-Gram Analysis for Training-Free Detection of GPT-Generated Text. *arXiv preprint arXiv:2305.17359*, 2023.

[Yang *et al.*, 2023b] Xianjun Yang, Liangming Pan, Xuandong Zhao, Haifeng Chen, Linda Petzold, William Yang Wang, and Wei Cheng. A Survey on Detection of LLMs-Generated Content. *arXiv preprint arXiv:2310.15654*, 2023.

[Yoo *et al.*, 2023] KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2092–2115, Toronto, Canada, July 2023.

[Yu *et al.*, 2023] Xiao Yu, Yuang Qi, Kejiang Chen, Guoqiang Chen, Xi Yang, Pengyuan Zhu, Weiming Zhang, and Nenghai Yu. GPT Paternity Test: GPT Generated Text Detection with GPT Genetic Inheritance. *arXiv preprint arXiv:2305.12519*, 2023.

[Zhan *et al.*, 2023] Haolan Zhan, Xuanli He, Qiongkai Xu, Yuxiang Wu, and Pontus Stenetorp. G3Detector: General GPT-Generated Text Detector. *arXiv preprint arXiv:2305.12680*, 2023.

[Zheng *et al.*, 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.