Does DETECTGPT Fully Utilize Perturbation? Bridge Selective Perturbation to Fine-tuned Contrastive Learning Detector would be Better

Anonymous ACL submission

Abstract

The burgeoning generative capabilities of large language models (LLMs) have raised growing concerns about abuse, demanding automatic machine-generated text detectors. DetectGPT (Mitchell et al., 2023), a zero-shot metric-based detector, first introduces perturbation and shows great performance improvement. However, in DetectGPT, random perturbation strategy could introduce noise, and logit regression depends on threshold, harming the generalizability and applicability of individual or small-batch inputs. Hence, we propose a novel fine-tuned detector, PECOLA, bridging metric-based and fine-tuned detectors by contrastive learning on selective perturbation. Selective strategy retains important tokens during perturbation and weights for multi-pair contrastive learning. The experiments show that PECOLA outperforms the state-of-the-art by 1.20% in accuracy on average on four public datasets. And we further analyze the effectiveness, robustness, and generalization of the method.

1 Introduction

Machine-generated text (MGT) detection is to discriminate MGT from human-written texts (HWT), preventing abuse of large language models (LLMs), including academic misconduct (Vasilatos et al., 2023), spam synthesis (Dou et al., 2020), untrustworthy news (Zellers et al., 2019), etc. Currently, existing MGT detection methods can be mainly classified into two genres (Wu et al., 2023), *i.e.*, fine-tuned methods (Liu et al., 2023; Hu et al., 2023; Verma et al., 2023; OpenAI, 2023) and zero-shot metric-based methods (Gehrmann et al., 2019; Mitchell et al., 2023; Yang et al., 2023; Bao et al., 2024). In general terms, fine-tuned detector methods can achieve better accuracy than zero-shot metric-based methods, especially generalizable to black-box generators, but are more costly during data collection, fine-tuning, and running. On the other hand, zero-shot metric-based methods show better interpretability than fine-tuned ones.

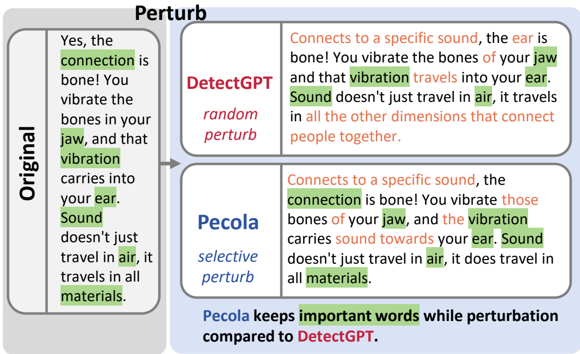

Figure 1: Example of the selective strategy perturbation of PECOLA, which prevent modifying important tokens (in green). Orange tokens are the perturbed texts.

DetectGPT (Mitchell et al., 2023), as an unsupervised zero-shot metric-based method, first introduces perturbation in MGT detection. Specifically, it applies random masking to the original input sample and uses T5 (Raffel et al., 2020) to fill in. It posits that minor perturbations of MGT tend to have lower log probability under the base model than the original sample. The introduction of perturbation in DetectGPT surpasses the vanilla log-probability-based method (Gehrmann et al., 2019) in white-box settings.

However, DetectGPT still has three significant defects: (*i*) DetectGPT's reliance on the logit regression module's threshold compromises its generalization in zero-shot settings and limited to large batch input, fails on individual inputs. (*ii*) DetectGPT does not fully utilize the perturbation. As a metrics-based method, it only considers the probability difference caused by perturbation, which is overly simplified and slightly indistinguishable. Perturbation should indeed be a stronger augment that carries implicit language pattern information. (*iii*) DetectGPT perturbs the original sample randomly and unrestricted, which could introduce more noise and negatively impact the performance

(Kim et al., 2022). For example, Liu et al. (2023) find entity-relationship plays a role in the detection, which might be destroyed in random perturbation of DetectGPT.

In this paper, we thus propose a **Pe**rturbation-based **Co**ntrastive **Lea**rning model, PECOLA, for MGT detection, toward the defects with two stages, *i.e.*, Selective Strategy Perturbation (Sec. 3.1) and Token-Level Weighted Multi-Pairwise Contrastive Learning (Sec. 3.2). **Firstly**, Selective Strategy Perturbation is a token-level rewriting method with restrictions on modifying important texts (Campos et al., 2020) to reduce noise. The motivation is to simulate the human behavior of modification (Verma and Lee, 2017; Fetaya et al., 2020; Wang et al., 2019). The perturbation strategy consists of tokens removing and substitution, as shown in Fig. 1. The experiments show that our Selective Strategy Perturbation method can improve the performance of both metrics-based (*i.e.*, DetectGPT) and model-based methods. **Secondly**, we use a Multi-Pairwise Contrastive Learning model to process the perturbed texts. Different from the logit regression module in DetectGPT, the trained model is generalizable without any threshold setting, and it can deal with individual inputs. Moreover, by utilizing multi-pairwise contrastive learning, the model could better utilize perturbation to focus on the language pattern gap between HWT and MGT. The importance weight from the perturbation stage is also reused as contrastive learning weight. Notably, by using contrastive learning, PECOLA is a strong few-shot fine-tuning method, which effectively bridges and integrates metric-based and fine-tuned detector genres. **Finally**, extensive experiments show PECOLA is significantly superior to baseline and SOTA methods on four datasets, PECOLA improves by 1.20% to SOTA on average under few-shot settings, surpassing the latest methods by 3.84% among metric-based detectors and by 1.62% among fine-tuned detectors. Further experiments show that PECOLA is as well better at generalization, robustness, and effectiveness.

Our contributions are summarized as follows:

- **Selective Perturbation**: Based on our analysis of various selective perturbation strategies, we propose a novel method considering token importance, which reduces the noise and benefits to both supervised and unsupervised approaches.
- **Bridge Metric and Model-based Detectors**: We utilize a novel fine-tuned contrastive learning module to replace the logit regression of DetectGPT (metric-based), which frees the detector from setting the threshold, enables it to deal with individual input, and can be generalizable and effective on the few-shot setting by contrasting perturbed texts with origin ones.
- **Outperformance**: Our detector PECOLA outperforms all eight compared models on four public datasets. And PECOLA is more robust to the choice of base model and filling model. Furthermore, we prove its generalization ability across domains and genres of data.

2 Related Work

Machine-generated Text Detection. While fine-tuned detectors have proven effective for MGT detection (Wahle et al., 2022; Hu et al., 2023), the requirement for annotated datasets poses a significant challenge due to the proliferation of unchecked, high-quality generated texts. To address this challenge, DetectGPT (Mitchell et al., 2023) and Fast-DetectGPT (Bao et al., 2024) have demonstrated strong performance in white-box zero-shot settings. Similarly, CoCo (Liu et al., 2023) is designed to detect MGT with low resource annotations, utilizing a coherence-based contrastive learning model. Recently, the Watermark method (Kirchenbauer et al., 2023) has been introduced to mitigate the risk associated with unchecked MGTs by embedding imperceptible signals within text outputs. In contrast to previous methods, our approach integrates data perturbation with contrastive learning, with a particular emphasis on reducing reliance on mask-filling models and enhancing performance in few-shot scenarios.

Perturbation. Data perturbation methods find frequent application in text classification tasks (Gao et al., 2022; Shum et al., 2023), which is commonly employed through the technique of consistency regularization (Xie et al., 2020; Chen et al., 2020). Nevertheless, in MGT detection, previous perturbation methods have exhibited certain limitations. For instance, they often resort to randomly selecting target tokens for synonym replacement (Wang et al., 2018), deletion, or insertion (Wei and Zou, 2019), and fine-tuning pre-trained models to fill text spans of variable lengths (Gao et al., 2022). While these methods do enhance text diversity, the indiscriminate replacement of tokens without guided rules can lead to the generation of less reliable texts.

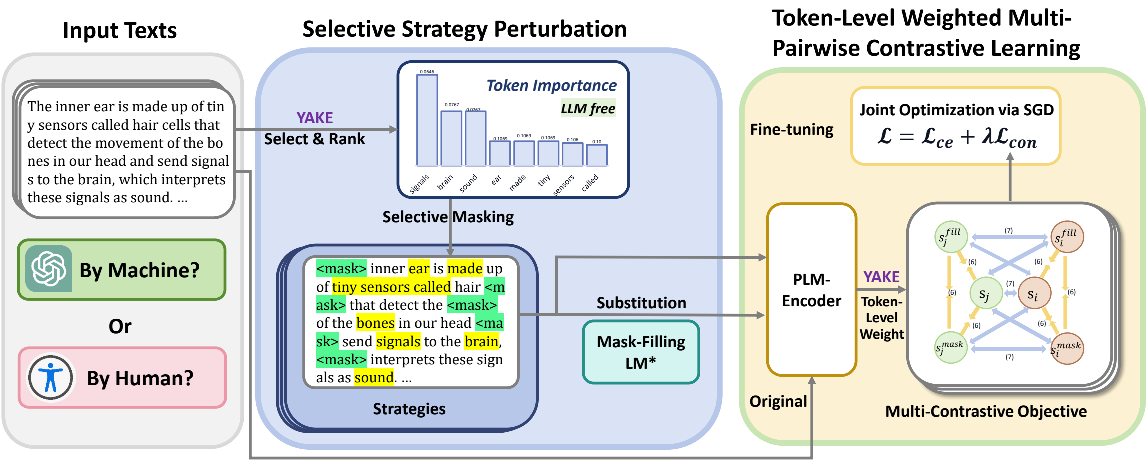

Figure 2: Overview of PECOLA. In the Selective Strategy Perturbation stage (Sec. 3.1), we use the YAKE algorithm to score token importance and then selective masking based on probability. Then, we fill in the masks with a mark-filling language model. In the Contrastive Learning stage (Sec. 3.2), we design a multi-pairwise method with token-level weights also from tokens importance. Yellow arrows represent attraction and green ones represent repulsion. The model is optimized by combining cross-entropy (CE) loss $\mathcal{L}_{\text{ce}}$ and contrastive loss $\mathcal{L}_{\text{Margin}_w}$. * Our method, different from DetectGPT, is generalizable on any mask-filling language model.

These limitations motivate us to devise data perturbation methods tailored for MGT detection. Our approach, guided selectively, aims to better represent meaningful recombination spaces while preserving the inherent semantic features of the text, ultimately enhancing the diversity of samples.

Constrastive Learning. Contrastive learning is an effective solution method to the issues of relying solely on cross-entropy classification loss leading to a lack of robustness and suboptimal generalization (Tack et al., 2020; Hu et al., 2023). In limited labeled data task (Gunel et al., 2021), introduce a robust contrastive learning method to capture the similarities between the same instances in the representation space, while separating those of different classes. Similarly, out-of-distribution (OOD) usually leads to severe semantic shift issues during inference, prompting another approach based on margin contrastive learning (Zhou et al., 2021). Differently, our method focuses more on the changes of the rephrase space in data distribution after perturbation, and strives to reduce reliance on the mask-filling models in few-shot learning.

3 Methodology

As shown in Fig. 2, the workflow of PECOLA mainly consists of two stages: Selective Strategy Perturbation and Supervised Contrastive Learning for fine-tuning PLM, which enjoy the advantage of metric-based and model-based detectors, respectively.

3.1 Selective Strategy Perturbation

In this work, we present a token-level selective strategy perturbation method to relieve the information loss caused by the random masking used in DetectGPT. Our approach involves adapting the mask-selection probability for each text token based on its importance, thus generating perturbed inputs with strategically placed masks. Additionally, we harness LLMs to populate the masks, creating filled perturbation inputs. This step effectively introduces a diverse range of perturbation information into our detection model.

Token Importance Assessment. To accurately assess the significance of tokens within the text and mitigate information loss stemming from random masking, we expand upon the YAKE algorithm (Campos et al., 2020) to operate at the token level. The YAKE algorithm builds upon certain assumptions (Machado et al., 2009), which posit that the importance of a candidate word decreases as the richness of the vocabulary surrounding it on both sides increases. This fundamental assumption remains applicable when processing text at the token level, *i.e.*, token importance assessment.

Specifically, considering a training set S comprising i inputs, for each text input $s_i \in S$ containing n tokens (*i.e.*, $s_i = \{e_i^1, e_i^2, \ldots, e_i^n\}$), we employ the YAKE algorithm to compute a score for each token e. Tokens with scores falling below the specified threshold α are then incorporated into the set of important tokens K_i:

$$K_i = \begin{cases} K_i \cap \{e_i^n\}, & \text{if } Score(e_i^n) < \alpha \\ K_i, & \text{otherwise} \end{cases} \quad (1)$$

where $Score(e_i^n)$ represents the YAKE score calculated by token e_i^n. The higher the score, the lower the importance of the token e_i^n in s_i.

Mask Position Selection. After getting the important tokens set K_i of each text input s_i, we use special token `[MASK]` to replace some of the tokens in the text input to construct masked perturbation input s_i^{mask}. In order to relieve the information loss caused by masking perturbation, we add regularization to the traditional random masking method and use a selective masking strategy to prevent important tokens from being masked.

Given an input text $s_i = \{e_i^1, e_i^2, \ldots, e_i^n\}$, we use the selective masking strategy to traverse each token and determine whether to mask it based on the token's importance. The probability of token e_i^n being masked is specifically defined as:

$$P_i^n = \mathbf{1}_{[e_i^n \in K_i]} P \quad (2)$$

where P is the mask ratio, and $\mathbf{1}_{[e_i^n \in K_i]}$ represents an indicator function with a value of 1 if and only if the condition $e_i^n \in K_i$ is satisfied, otherwise, it is 0. Then we gather all masked perturbation inputs $\{s_1^{mask}, ..., s_i^{mask}\}$ and include them in the training set to give the model masked perturbation improving model robustness.

Mask-Filling. Additionally, we utilize the current PLMs, *e.g.*, T5 (Raffel et al., 2020) or RoBERTa (Liu et al., 2019) etc., to fill the masked perturbation inputs and create the filled perturbation inputs $\{s_1^{fill}, \ldots, s_i^{fill}\}$. Similar to before, we include all filled perturbation inputs in the training set and obtain the final training set $S = \{s_1, \ldots, s_i, s_1^{mask}, \ldots, s_i^{mask}, s_1^{fill} \ldots, s_i^{fill}\}$.

3.2 Token-Level Weighted Multi-Pairwise Contrastive Learning

Importance-based Feature Reconstruction. Existing MGT methods (Liu et al., 2023) often uniformly extract all token information in the text, ignoring the huge impact of a few important tokens on the detection model. In this work, we reconstruct the token feature extracted by PLM according to the importance of the token in the input text, allowing the detection model to focus more on important token information. We assign adaptive weights to all tokens in the input:

$$w_i^n = \begin{cases} 1 - Score(e_i^n), & \text{if } e_i^n \in K_i \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where w_i^n represents the assign adaptive weight of the n-th token of the i-th input in the training set. After that, we use the last hidden layer embedding of the outputs in the base PLMs to extract input features:

$$H_i = \text{PLM}(s_i) \quad (4)$$

where H_i contains the features of all tokens in the input s_i, *i.e.*, $H_i = \{h_i^1, h_i^2, \ldots, h_i^n\}$. We use the weight of the corresponding token to reconstruct its features:

$$h_i^n = h_i^n(1 + w_i^n) \quad (5)$$

By using feature reconstruction, we assign more weight to important tokens. This allows our detection model to concentrate on the characteristic information of these important tokens.

Multi-Pairwise Contrastive Learning. Considering that existing works (Gunel et al., 2021; Zhou et al., 2021; Liu et al., 2023) mainly concentrate on single-input feature learning while overlooking input correlations, we introduce contrastive learning into MGT. It enables PECOLA to discern the distinct featurinputes of variously labeled data, more accurately capture input features, and significantly enhance performance in few-shot setting.

Given a batch training data $\{s_i\}_{i=1}^M$, where M is the batch size, we calculate the positive class contrastive loss and negative class contrastive loss on the last hidden layer embedding of the first token output h_i^1 from the base PLM:

$$\mathcal{L}_{\text{pos}} = \sum_{i=1}^{M} \frac{1}{|P_t(i)|} \sum_{p \in P_t(i)} \|(h_i^1 - h_p^1)\|^2 \quad (6)$$

$$\mathcal{L}_{\text{neg}} = \sum_{i=1}^{M} \frac{1}{|N_t(i)|} \sum_{n \in N_t(i)} \max\left(0, \xi - \|(h_i^1 - h_n^1)\|^2\right) \quad (7)$$

where $P_t(i)$ represents the samples with the same label as the i-th sample in the batch, and $N_t(i)$ represents the ones with different labels as the i-th sample. And ξ is the maximum L_2 distance between pairs of inputs from the same class in the batch of training data:

$$\xi = \max_{i=1}^{M} \max_{p \in P_t(i)} \|h_i^1 - h_p^1\|^2 \quad (8)$$

This adaptive margin ensures that despite data Perturbation during training, the model is steered to maintain discriminative embeddings. Then we get the following contrastive loss as:

$$\mathcal{L}_{\text{con}} = \frac{1}{M}(\mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{neg}}) \tag{9}$$

For supervised learning tasks, we utilize the cross-entropy classification loss $\mathcal{L}_{ce}$ to train our detection model. By adjusting the weight λ to balance the impact of various losses on the model, our total loss is given by the following:

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \lambda\mathcal{L}_{\text{con}} \tag{10}$$

4 Experiments

4.1 Experiment Settings

To demonstrate the effectiveness of PECOLA, we conduct extensive experiments on four open-source datasets under few-shot learning settings.

Datasets. Grover (Zellers et al., 2019), generated by the transformer-based news generator Grover-Mega (1.5B); **GPT-2**, a webtext dataset provided by OpenAI (2019) based on GPT-2 XL (1.5B); **GPT-3.5**, a news-style dataset constructed by CoCo (Liu et al., 2023) using the text-DaVinci-003 model (175B); **HC3** (Guo et al., 2023), involving open domains, finance, healthcare, law, and psychology texts, composed of comparative responses from human experts and ChatGPT.

Few-shot Learning Settings. We randomly sample 32, 64, 128 and 512 samples from the original training set, while keeping the balance of machine and human categories. More details are provided in Appendix A.1.

4.2 Comparison Models

we compare PECOLA with both unsupervised and supervised MGT detection methods:

RoBERTa (Liu et al., 2019), supervised methods via standard fine-tuning pre-trained language models as classifiers. We use RoBERTa-base (135M).

GLTR (Gehrmann et al., 2019), a metric-based detector and based on next-token probability. We follow the setting of Guo et al. (2023), utilizing the Test-2 feature. For a fair comparison with fine-tuning methods, we first use the few-shot training samples to settle the threshold and adapt the fixed threshold in the test set.[1]

CE+SCL (Gunel et al., 2021), a fine-tuned detector, used in conjunction with the Cross-Entropy (CE) loss, exhibiting impressive performance in few-shot learning settings.

CE+Margin (Zhou et al., 2021), a contrastive learning approach focuses on separating OOD instances from In-Distribution (ID) instances, aiming to minimize the L_2 distance between instances of the same label. We train the detector by combining CE loss.

IT:Clust (Shnarch et al., 2022), a general text classification method that employs unsupervised clustering as an intermediate for fine-tuning pre-trained models, utilizing RoBERTa-base (110M).

CoCo (Liu et al., 2023) utilizes coherence graph representation and contrastive learning to improve supervised fine-tuning methods in both inadequate and adequate data resource scenarios.

DetectGPT (Mitchell et al., 2023), a zero-shot metric-based MGT detector, using T5-large (Raffel et al., 2020) to perturb texts. Same as *GLTR*, we fix the threshold.[2]

Fast-DetectGPT (Bao et al., 2024), an optimized zero-shot detector, building upon the foundation of DetectGPT, and utilizes a surrogate GPT-Neo (2.7B) (Black et al., 2022) model for scoring.

4.3 Performance Comparison

As shown in Table 1, PECOLA surpasses the competitors on all datasets in the few-shot MGT detection task. Specifically, compared with the best competitor, PECOLA achieves accuracy and F1-score improvement of 2.04% and 1.42%, 1.71% and 2.55% on Grover and GPT2 datasets. On GPT3.5 and HC3 datasets, PECOLA still ensures 0.86% and 0.68%, 0.21% and 0.22% performance improvement with greater stability. The results prove the effectiveness of PECOLA, which integrates the advantage of unsupervised (perturbation for metric-based) and supervised (contrastive learning for model-based) MGT detection methods.

Moreover, the unsupervised learning methods tend to show better performance in extremely few shot scenarios. Unsurprisingly, unsupervised methods do not see a notable performance improvement with the increase in the number of training samples, which causes them to outperform on the fewest shot settings initially but soon be surpassed. As for the

[1] The base model of *GLTR* is chosen based on the generator of the dataset: for GPT-2 and Grover datasets, we use GPT-2 Small (124M); and for GPT-3.5 and HC3 datasets, we use GPT-J (6B) (Wang, 2021), which is the best open-source model to simulate ChatGPT and GPT-3.5 empirically.

[2] For all four datasets (including HC3 and GPT-3.5 datasets), we use GPT-2 Small (124M) as the base model to calculate the likelihood. The reason is Mireshghallah et al. (2023) find that small model is better black-box detector for *DetectGPT*.

Dataset	Metric	Shot	*RoBERTa*	*GLTR*†	*CE+SCL*	*CE+Margin*	*IT:Clust*	*CoCo**	*DetectGPT*†*	*Fast-Detect.*†*	PECOLA
Grover	*Acc*	32	$48.83_{10.31}$	56.61	$55.86_{4.43}$	$56.79_{3.31}$	$41.57_{3.58}$	$51.60_{8.42}$	55.02	56.06	$\mathbf{59.03_{1.63}}$
		64	$56.88_{3.03}$	56.61	$57.57_{2.63}$	$58.92_{2.17}$	$46.45_{2.20}$	$58.27_{10.21}$	54.61	60.33	$\mathbf{60.94_{1.56}}$
		128	$59.28_{1.91}$	58.48	$60.33_{3.41}$	$60.44_{3.85}$	$50.72_{3.70}$	$58.97_{5.53}$	55.78	60.33	$\mathbf{63.60_{1.71}}$
		512	$70.39_{1.21}$	62.26	$72.38_{1.73}$	$72.15_{1.16}$	$56.08_{0.87}$	$70.07_{5.54}$	55.56	62.50	$\mathbf{73.12_{0.84}}$
	F1	32	$44.13_{8.82}$	52.77	$51.56_{3.03}$	$53.21_{2.24}$	$40.79_{3.66}$	$47.33_{2.63}$	51.09	56.67	$\mathbf{53.95_{0.94}}$
		64	$52.88_{1.52}$	52.77	$53.39_{1.16}$	$54.99_{1.75}$	$46.10_{1.25}$	$44.70_{3.53}$	48.07	57.92	$\mathbf{55.48_{1.35}}$
		128	$54.69_{1.18}$	54.47	$55.74_{2.21}$	$55.54_{2.40}$	$51.37_{4.80}$	$51.44_{2.13}$	53.78	54.89	$\mathbf{58.98_{1.58}}$
		512	$64.49_{3.17}$	57.11	$67.02_{2.12}$	$66.25_{1.65}$	$51.80_{0.49}$	$65.15_{3.76}$	53.32	61.29	$\mathbf{68.24_{1.64}}$
GPT-2	*Acc*	32	$70.53_{4.10}$	**75.99**	$69.32_{5.19}$	$70.00_{2.33}$	$51.02_{1.66}$	$71.69_{7.07}$	68.59	71.88	$75.42_{1.80}$
		64	$74.41_{2.47}$	75.76	$73.77_{3.54}$	$74.04_{1.42}$	$54.32_{2.73}$	$73.20_{1.42}$	71.12	71.88	$\mathbf{78.92_{1.14}}$
		128	$79.77_{2.04}$	75.77	$80.18_{1.25}$	$80.93_{1.26}$	$59.66_{2.83}$	$79.44_{4.80}$	71.74	71.88	$\mathbf{82.58_{0.49}}$
		512	$84.07_{1.46}$	75.86	$84.76_{1.19}$	$84.89_{1.17}$	$71.59_{3.23}$	$84.30_{0.58}$	71.74	74.06	$\mathbf{85.75_{0.69}}$
	F1	32	$66.57_{5.09}$	72.45	$64.89_{8.13}$	$69.89_{2.38}$	$48.45_{3.72}$	$71.19_{11.05}$	65.50	70.00	$\mathbf{75.10_{1.99}}$
		64	$73.91_{2.69}$	70.87	$72.32_{4.31}$	$73.94_{1.40}$	$53.87_{3.00}$	$69.79_{2.03}$	66.58	70.97	$\mathbf{78.88_{1.17}}$
		128	$79.49_{2.26}$	71.16	$80.00_{1.35}$	$80.79_{1.34}$	$59.48_{2.79}$	$76.10_{7.37}$	66.13	71.88	$\mathbf{82.54_{0.51}}$
		512	$84.01_{1.52}$	75.56	$84.72_{1.25}$	$84.86_{1.24}$	$70.42_{4.26}$	$83.88_{0.79}$	66.13	74.64	$\mathbf{85.72_{0.70}}$
GPT-3.5	*Acc*	32	$90.54_{7.26}$	92.55	$92.44_{3.19}$	$92.85_{2.44}$	$61.82_{4.30}$	$93.27_{1.44}$	84.42	89.10	$\mathbf{95.80_{0.68}}$
		64	$96.85_{0.84}$	91.00	$96.86_{1.67}$	$97.32_{0.58}$	$77.70_{6.92}$	$95.76_{1.52}$	82.58	89.65	$\mathbf{98.01_{0.31}}$
		128	$97.50_{1.24}$	91.60	$98.00_{0.46}$	$98.00_{0.18}$	$92.54_{4.01}$	$96.26_{0.89}$	85.33	89.85	$\mathbf{98.06_{0.12}}$
		512	$98.97_{0.18}$	92.60	$98.99_{0.80}$	$98.92_{0.28}$	$98.13_{1.20}$	$98.05_{0.47}$	85.57	90.62	$\mathbf{99.14_{0.15}}$
	F1	32	$90.27_{7.77}$	92.71	$92.42_{3.20}$	$92.81_{2.49}$	$60.95_{4.67}$	$92.72_{1.54}$	84.43	89.76	$\mathbf{95.80_{0.68}}$
		64	$96.84_{0.84}$	91.49	$96.86_{1.67}$	$97.47_{0.30}$	$77.33_{7.31}$	$95.45_{1.54}$	86.16	89.92	$\mathbf{98.01_{0.31}}$
		128	$97.50_{1.24}$	91.96	$98.00_{0.46}$	$98.00_{0.18}$	$92.50_{4.07}$	$97.57_{0.92}$	86.13	89.77	$\mathbf{98.06_{0.12}}$
		512	$98.85_{0.40}$	92.71	$98.93_{0.21}$	$98.92_{0.28}$	$98.13_{1.20}$	$97.88_{0.50}$	86.20	90.62	$\mathbf{99.14_{0.15}}$
HC3	*Acc*	32	$93.36_{1.50}$	**97.30**	$95.33_{1.81}$	$95.46_{1.71}$	$77.00_{8.05}$	$92.11_{1.71}$	94.54	87.70	$97.19_{0.16}$
		64	$96.97_{0.74}$	98.13	$97.81_{0.41}$	$97.81_{0.31}$	$91.69_{2.34}$	$95.50_{1.27}$	95.03	88.87	$\mathbf{98.59_{0.14}}$
		128	$97.56_{0.38}$	98.29	$98.17_{0.30}$	$98.14_{0.36}$	$95.43_{1.15}$	$97.57_{1.09}$	95.10	88.87	$\mathbf{98.63_{0.32}}$
		512	$98.85_{0.40}$	98.31	$98.93_{0.21}$	$98.99_{0.20}$	$97.98_{0.47}$	$98.58_{1.18}$	95.13	90.62	$\mathbf{99.15_{0.11}}$
	F1	32	$93.34_{1.52}$	**97.30**	$95.32_{1.82}$	$95.45_{1.72}$	$76.47_{8.77}$	$92.07_{1.56}$	94.29	88.39	$97.19_{0.16}$
		64	$96.97_{0.74}$	98.12	$97.81_{0.41}$	$97.81_{0.32}$	$91.67_{2.34}$	$95.50_{1.19}$	94.95	89.92	$\mathbf{98.59_{0.14}}$
		128	$97.56_{0.38}$	98.29	$98.17_{0.30}$	$98.14_{0.36}$	$95.43_{1.15}$	$97.59_{1.05}$	95.01	89.92	$\mathbf{98.63_{0.32}}$
		512	$98.85_{0.40}$	98.31	$98.93_{0.21}$	$98.99_{0.20}$	$97.98_{0.47}$	$98.59_{1.16}$	95.05	91.06	$\mathbf{99.15_{0.11}}$

Table 1: Comparison of PECOLA to baseline methods in few-shot MGT detection. The results are average values of 10 runs with different random seeds. The subscript means the standard deviation (*e.g.*, $99.15_{0.11}$ means 99.15 ± 0.11). † Zero-shot model-based methods' results are deterministic, so we do not report standard deviation. Also, these methods must have the white-box generator as the base model, which is different from the black-box settings of other model-based methods. Asterisk (*) denotes the latest SOTA method.

deception of generators, Grover appears to be the hardest to detect while other models are relatively "honest" to detectors. It might have originated from the adversarial training strategy of Grover, while the bulit-in detector module adversarially shifts the LLM's detectable features. More interestingly, advanced language models show a weaker ability to cheat detectors. Most detectors achieve around 98% in accuracy on the GPT-3.5 and HC3 datasets, which is consistent with the conclusion from Liu et al. (2023); Chen et al. (2023). We hypothesize that the easy-to-detect nature may originate from the lack of semantics diversity in GPT-3.5 and ChatGPT as they are using RLHF (Kirk et al., 2023).

4.4 Ablation Study

To illustrate the effectiveness of the PECOLA components, we study the ablation experiments on the Selective Strategy Perturbation stage and the Contrastive Learning stage on the 64-example GPT-2 dataset. We also demonstrate the Scalability of PECOLA in Appendix C.

Method	*Acc*	*F1*
w/o. mask	$78.00_{1.40}$	$77.93_{1.43}$
w/o. mask-fill	$77.78_{1.82}$	$77.72_{1.83}$
w/o. mask.CL_w	$75.80_{2.22}$	$75.23_{2.46}$
w/o. mask-fill. CL_w	$75.56_{1.47}$	$75.10_{1.73}$
w/o. CL_w	$76.60_{1.69}$	$76.22_{1.65}$
w/o. w	$78.02_{1.56}$	$77.93_{1.57}$
PECOLA	$78.92_{1.14}$	$78.88_{1.17}$

Table 2: Ablation study result of PECOLA.

Ablation on Selective Strategy Perturbation. In PECOLA, the data used for training primarily includes original texts, selected mask texts, and mask-filled texts. We remove each part of the data in training, *i.e.*, (*i*) **w/o. mask**, refers to not using selected mask texts for training; (*ii*) **w/o. mask-fill**,

Model	**RoBERTa**		**PECOLA**	
Metric	*Acc*	*F1*	*Acc*	*F1*
Original	$74.41_{2.47}$	$73.91_{2.69}$	$78.92_{1.14}$	$78.88_{1.17}$
Delete	$71.77_{5.88}$(-2.640)	$70.42_{8.05}$(-3.490)	$77.28_{1.79}$(-1.640)	$77.06_{2.03}$(-1.820)
Repeat	$64.69_{6.63}$(-9.720)	$61.74_{9.20}$(-12.17)	$69.74_{4.83}$(-9.180)	$67.87_{6.24}$(-11.01)
Insert	$50.75_{0.67}$(-23.66)	$36.44_{1.60}$(-37.47)	$57.61_{2.52}$(-21.31)	$49.29_{4.57}$(-29.59)
Replace	$52.04_{1.58}$(-22.37)	$39.48_{3.59}$(-34.43)	$57.25_{2.21}$(-21.67)	$48.89_{3.93}$(-29.99)
Average	59.81 (-14.60)	52.02 (-21.89)	65.47 (-13.45)	60.78 (-18.10)

Table 3: Model robustness to four perturbations.

Model	**RoBERTa**	**PECOLA**
Random Mask **DetectGPT**	-2.64	-1.64
Selective Mask **PECOLA**	-0.72	-1.15
Mask-Filling **DetectGPT**	-4.72	-2.66
Mask-Filling **PECOLA**	-1.34	-1.34

Table 4: Affinity of DetectGPT's and PECOLA's masking strategy on RoBERTa and PECOLA.

not using mask-filling texts for training.

Ablation on Contrastive Learning. It primarily investigates the impact of CE and contrastive loss. (*i*) **w/o. CL**$_w$ refers to the model larking weighted contrastive learning; (*ii*) **w/o.** w refers to the model including contrastive learning but larking weight.

As demonstrated in Table 2, in scenarios employing only the CE loss, the Selective Strategy Perturbation method contributes to significant performance improvement. Moreover, the introduction of weighting further enhances accuracy when compared to the direct use of margin loss. It reveals the validation of bridging the metric-based and model-based detectors, *i.e.*, employing the Selective Strategy Perturbation method to evaluate the token importance for the multi-pairwise contrastive learning method. Furthermore, within the overall framework, the removal of the select mask text results in a more rapid decrease in accuracy compared to the removal of the mask-filling text. This finding substantiates that the Token-Level Weighted Multi-Pairwise Contrastive Learning method can better focus on the alterations in the rephrased space following the application of Selective Strategy Perturbation to the text.

4.5 Discussion and Analysis

4.5.1 Model Qualities

We analyze the model qualities, including robustness and affinity in this section. Here, we test on the 10,000-example GPT-2 test dataset, and the perturbation scale is set to 15%.

Analysis on Robustness. To validate the robustness of PECOLA in the few-shot learning settings, we apply four post hoc perturbation operations for each token in the test dataset randomly, *i.e.*, deletion, replacement, insertion, and repetition. As indicated in Table 3, for each perturbation method employed, our decline rate is consistently lower compared to the baseline RoBERTa. On average, PECOLA maintains a 5.66% higher accuracy and an 8.77% superior F1-score. Specifically, in the deletion method, where we introduce a 15% random perturbation, it is noteworthy that the accuracy of PECOLA utilizing the Selective Strategy Perturbation method experiences a mere 1.64% decrease, underscoring its remarkable robustness.

Analysis on Affinity. Affinity pertains to alterations in data distribution resulting from perturbations, quantified by observing the fluctuations in accuracy. We demonstrate the superiority of the selective masking method over the random masking method using the Affinity metric, following the setting of DetectGPT. We applied a 15% mask proportion with a span of 2 tokens on the test dataset and simultaneously employed T5-Large (Raffel et al., 2020) as the mask-filling model. We trained RoBERTa-base and PECOLA on the 64-example GPT2 dataset. As shown in Table 4, in comparison to the random masking perturbation method utilized in DetectGPT, we observe a 1.92% and 0.49% increase in Affinity when employing the selective masking method. Additionally, the mask-filling method yields affinity improvements of 3.38% and 1.32% for RoBERTa and PECOLA models, respectively. These results illustrate that the Selective Multi-Strategy Perturbation method introduced in this paper effectively preserves more distinguishable features between MGTs and HWTs.

4.5.2 Analysis on Selective Strategies

Our method is a neat perturbation strategy that is mainly for the MGT detection task and empirically shows great performance. Beyond the PECOLA and random perturbation method, we experiment with two other perturbation strategies: rank-based perturbation and keyword-based perturbation. In rank-based perturbation, we use the rescaled rank of next-token probability on GPT2-medium as the weight for perturbation position selection. In keyword-based perturbation, we prevent changes in the keywords extracted by the VLT-5 language model (Pęzik et al., 2022) during perturbation. As shown in Table 5, the experimental results of selective perturbation outperform the random pertur-

Method	Random	Prob. Rank	Keyword	Importance
Yake	$76.05_{1.83}$	$77.35_{0.73}$	$78.55_{1.65}$	$\mathbf{78.92_{1.14}}$
Perplexity	$75.53_{1.14}$	$76.63_{1.03}$	$77.11_{1.80}$	$\mathbf{77.63_{1.30}}$

Table 5: Different strategies for perturbation and token-Level weighting, namely Random (DetectGPT), Prob. Rank (GPT2-medium), Keyword (VLT-5), Importance (PECOLA).

Method	Random	Prob. Rank	Keyword	Importance
Ratio (%)	9.20	7.83	7.80	5.56

Table 6: Mask-filling failure ratio.

bation method by 1.20%, 2.04%, and 2.49% in average accuracy on the 64-example GPT2 dataset. The Importance-based strategy of PECOLA is the highest. We further find that the random strategies of DetectGPT lead to more masking-filling failures than selective ones, which cause execution errors. Mask-filling failure ratio in Table 6 indicates that selective strategy based on token importance greatly decreases the failure ratio by 3.64%.

4.5.3 Generalization on Mask-Filling Models

We study the influence of various mask-filling models on the performance of PECOLA, including Bert (110M; Devlin et al. 2019), Bart (139M; Mike et al. 2020), GPT-2 (380M; Radford et al. 2019), Twhin-bert (279M; Zhang et al. 2023), XLM (279M; Alexis et al. 2020), XLNet (110M; Yang et al. 2019), RoBERTa(135M; Liu et al. 2019), and LLaMa-2(7B; Touvron et al. 2023). As depicted in Fig. 3, the results of all mask-filling models, in accuracy, surpass the baseline. Furthermore, the fluctuation of PECOLA's performance across different mask-filling models is relatively slight. It confirms that PECOLA is not reliant on a specific filling model, showing great generalization capability. The experiment details and results are in Appendix D.1.

4.5.4 Generalization on Data

Cross-domain. We evaluate PECOLA on the HC3 dataset, the meta-information details are in Appendix A.2. For the three domains of data, we use one of them as training data (64-shot), and the remaining domains of data as testing data. The results in Table 7 show that PECOLA is more effective than the best baseline and SOTA method on average. For example, compared to Roberta, PECOLA outperforms by 4.61% in three domains on average. And PECOLA maintains a 1.63% higher accuracy on average than SOTA DetectGPT.

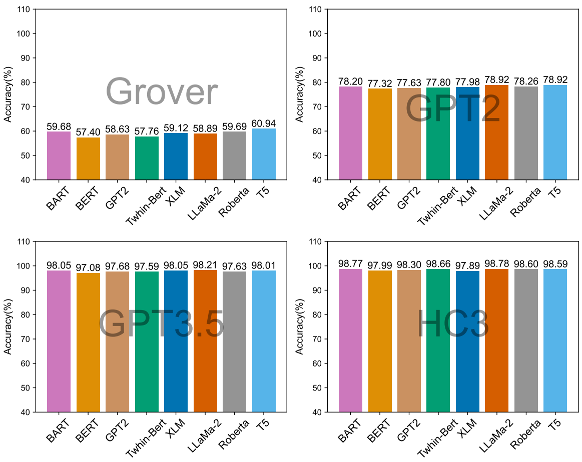

Figure 3: Result of using various mask-filling models.

Domain	Medicine	Finance	Computer Sci.	Average
RoBERTa	$62.97_{4.09}$	$86.08_{3.63}$	$90.64_{5.07}$	79.90
DetectGPT	**80.48**	85.17	82.98	82.88
PECOLA	$70.86_{7.83}$	$\mathbf{89.34_{2.93}}$	$\mathbf{93.32_{3.64}}$	**84.51**

Table 7: Results of cross-domain in terms of accuracy.

Cross-genre. We generalize PECOLA between News articles (GPT3.5 dataset) and QA answers (HC3 dataset) on the 64-shot settings. As shown in Table 8, when the GPT-3.5 dataset is the training set, PECOLA outperforms by 10.21%; and when the HC3 dataset is the training set, PECOLA outperforms by 6.98% to the best competitor.

Dataset	GPT3.5→HC3	HC3→GPT3.5	Average
RoBERTa	$64.60_{1.96}$	$62.67_{2.41}$	63.64
DetectGPT	77.11	72.66	74.89
PECOLA	$\mathbf{78.79_{8.19}}$	$\mathbf{72.87_{6.06}}$	**75.83**

Table 8: Results of cross-genre in terms of accuracy.

5 Conclusion

In this paper, we introduce PECOLA, a novel machine-generated text detection method that effectively bridges and integrates metric-based and fine-tuned detectors for MGT detection. To relieve the information loss caused by the random masking used in DetectGPT, we present a token-level selective strategy perturbation method. To more fully distinguish meaningful recombination spaces and reduce reliance on the mask-filling models, we present a token-level weighted multi-pairwise contrastive learning method. In few-shot settings, experimental results show that PECOLA significantly enhances the performance of PLMs in MGT detection. Subsequent analytical experiments validate PECOLA's effectiveness, robustness, generalization, and capability in detecting short texts.

Limitations

In this work, we focus on MGT detection in few-shot learning settings. The next phase will involve a more comprehensive performance comparison based on full datasets. Secondly, our method mentions the score threshold, if the threshold is too high or too low, it will not serve the purpose of perturbation. How to automate and flexibly design a strict threshold is also a direction for our next phase of improvement. Thirdly, for short texts, our perturbation method faces similar limitations, as it is difficult to extract the most relevant keywords. Thus, perturbation introduces more uncontrollable noise, which poses a challenge for us to address in the future. Fourth, We hope that the present work can inspire future applications in fields like machine-generated images and videos, creating a universal approach to apply in the direction of machine generation.

Ethics Statement

PECOLA aims to help users use our method to more reasonably and accurately identify MGT. Our goal is to develop a universal method applicable to other fields such as images and audio, and inspire the advancement of the stronger detector of MGTs and prevent all potential negative uses of language models. We do not wish our work to be maliciously used to counter detectors. The datasets mentioned in this paper are all public.

References

Conneau Alexis, Khandelwal Kartikay, Goyal Naman, Chaudhary Vishrav, Wenzek Guillaume, Guzmán Francisco, Grave Edouard, Ott Myle, Zettlemoyer Luke, and Stoyanov Veselin. 2020. Unsupervised cross-lingual representation learning at scale. In *Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *Computing Research Repository*, abs/2006.14799.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.

Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, and Bhiksha Ramakrishnan. 2023. Gpt-sentinel: Distinguishing human and chatgpt generated content. *arXiv preprint arXiv:2305.07969*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Yingtong Dou, Guixiang Ma, Philip S Yu, and Sihong Xie. 2020. Robust spammer detection by nash reinforcement learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 924–933.

Ethan Fetaya, Joern-Henrik Jacobsen, Will Grathwohl, and Richard Zemel. 2020. Understanding the limitations of conditional generative models. In *International Conference on Learning Representations*.

Jun Gao, Changlong Yu, Wei Wang, Huan Zhao, and Ruifeng Xu. 2022. Mask-then-fill: A flexible and effective data augmentation framework for event extraction. In *Conference on Empirical Methods in Natural Language Processing*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. 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.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *International Conference on Learning Representations*.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *arXiv preprint arXiv:2307.03838*.

Hazel H Kim, Daecheol Woo, Seong Joon Oh, Jeong-Won Cha, and Yo-Sub Han. 2022. Alp: Data augmentation using lexicalized pcfgs for few-shot text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10894–10902.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2023. Coco: Coherence-enhanced machine-generated text detection under low resource with contrastive learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16167–16188.

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

David Machado, Tiago Barbosa, Sebastião Pais, Bruno Martins, and Gaël Dias. 2009. Universal mobile information retrieval. In *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments: 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part II 5*, pages 345–354. Springer.

Lewis Mike, Liu Yinhan, Goyal Naman, Ghazvininejad Marjan, Mohamed Abdelrahman, Levy Omer, Stoyanov Ves, and Zettlemoyer Luke. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Fatemehsadat Mireshghallah, Justus Mattern, Sicun Gao, Reza Shokri, and Taylor Berg-Kirkpatrick. 2023. Smaller language models are better black-box machine-generated text detectors. *arXiv preprint arXiv:2305.09859*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *ICML 2023*.

OpenAI. 2019. Gpt-2 output dataset. Website.

OpenAI. 2023. Ai text classifier. Website.

Piotr Pęzik, Agnieszka Mikołajczyk-Bareła, Adam Wawrzyński, Bartłomiej Nitoń, and Maciej Ogrodniczuk. 2022. Keyword extraction from short texts with a text-to-text transfer transformer. *ACIIDS (Companion)*, 1716:530–542.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1:9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21:5485–5551.

Eyal Shnarch, Ariel Gera, Alon Halfon, Lena Dankin, Leshem Choshen, Ranit Aharonov, and Noam Slonim. 2022. Cluster & tune: Boost cold start performance in text classification. In *Annual Meeting of the Association for Computational Linguistics*, page 7639–7653.

KaShun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. *arXiv preprint arXiv:2302.12822*.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. 2020. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *Advances in neural information processing systems*, 33:11839–11852.

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

Christoforos Vasilatos, Manaar Alam, Talal Rahwan, Yasir Zaki, and Michail Maniatakos. 2023. Howkgpt: Investigating the detection of chatgpt-generated university student homework through context-aware perplexity analysis. *arXiv preprint arXiv:2305.18226*.

Rakesh Verma and Daniel Lee. 2017. Extractive summarization: Limits, compression, generalized model and heuristics. *Computación y Sistemas*, 21:787–798.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2023. Ghostbuster: Detecting text ghostwritten by large language models. *arXiv preprint arXiv:2305.15047*.

Jan Philip Wahle, Terry Ruas, Frederic Kirstein, and Bela Gipp. 2022. How large language models are transforming machine-paraphrase plagiarism. In *Conference on Empirical Methods in Natural Language Processing*, page 952–963.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax.

Sheng Wang, Jinjiao Lian, Yuzhong Peng, Baoqing Hu, and Hongsong Chen. 2019. Generalized reference evapotranspiration models with limited climatic data based on random forest and gene expression programming in guangxi, china. *Agricultural Water Management*, 221:220–230.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861.

Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. 2021. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. *arXiv preprint arXiv:2103.07552*.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Conference on Empirical Methods in Natural Language Processing*, pages 6381–6387.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F Wong, and Lidia S Chao. 2023. A survey on llm-gernerated text detection: Necessity, methods, and future directions. *arXiv preprint arXiv:2310.14724*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268.

Xianjun Yang, Wei Cheng, Linda Petzold, William Yang Wang, and Haifeng Chen. 2023. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. *arXiv preprint arXiv:2305.17359*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Xinyang Zhang, Yury Malkov, Omar Florez, Serim Park, Brian McWilliams, Jiawei Han, and Ahmed El-Kishky. 2023. Twhin-bert: A socially-enriched pre-trained language model for multilingual tweet representations at twitter. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5597–5607.

Wenxuan Zhou, Fangyu Liu, and Muhao Chen. 2021. Contrastive out-of-distribution detection for pre-trained transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111.

A Implementation Details

This part mentions the hyperparameter settings and meta-information of the HC3 dataset.

A.1 Hyperparameter Details

Experiments evaluating competitors and PECOLA follow the setting of CoCo (Liu et al., 2023). The hyperparameter settings of all the methods in the experiment as shown in Table 9. We randomly select 10 different seeds for experiments, and report average test accuracy and F1-score.

Parameter	Value
Training Epochs	30
Optimizer	AdamW
Learning rate	1e-5
Weight Decay	0.01
Batch Size	16
Mask Gap	2
Mask Proportion	10%
Score threshold	0.4
Pre-trained model	RoBERTa-base

Table 9: Implementation details of hyperparameters.

A.2 Meta-information

We evaluate PECOLA effectiveness from domains and genres on the HC3 dataset, which primarily includes Medicine, Finance, and Computer Science domain QA, as shown in Table 10.

Domain	Medicine	Finance	Computer Science
Size	2585	8436	1684

Table 10: Meta-information of the HC3 dataset.

B Effect of Hyperparameters

In PECOLA, the primary hyperparameters include the mask proportion, mask gap of perturbation, and score threshold. The perturbation proportion refers to the mask rate in the texts. The perturbation mask gap ensures that several tokens following a masked token remain unmasked, and score threshold to control the number of Most Relevant Keywords.

B.1 Perturbation Proportion and Mask Gap

We evaluate the impact of different perturbation ratios and mask gap on accuracy, and perform a minor scan in a few-shot learning settings with a set of mask proportions {5, 8, 10, 15, 17, 20} and mask gap {0, 1, 2, 3, 4, 5}, average the results for each combination of parameters. And a mask gap of 2 and a perturbation ratio of 10% achieve the maximum average values. As shown in Fig. 4, it is found that the combination of a mask gap of 2 and a mask proportion of 10% yielded the best results, on the 64-example GPT-2 dataset.

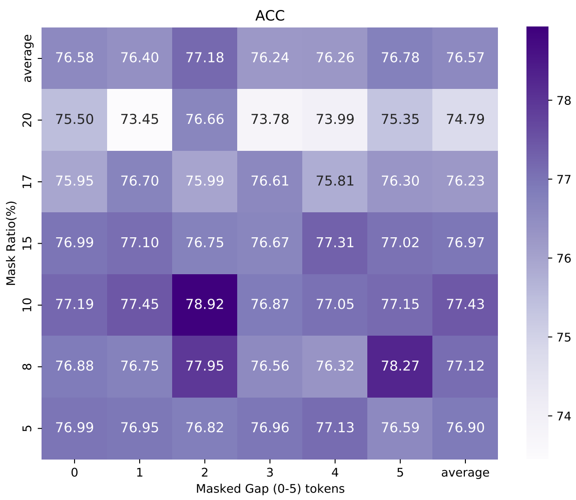

Figure 4: Impact of varying the number of perturbations and mask gap in PECOLA, we use T5-large (Raffel et al., 2020) as the mask-filling model. For each combination, we conduct tests on ten randomly select seeds.

B.2 Score Threshold

In the main experiment, all datasets use a common score threshold of 0.4, and it may not be the best choice for different datasets, because with the change in data type and text length, the gold keywords often vary. Therefore, as shown in Fig. 5, we discuss the performance changes of four datasets with different score threshold in few-shot learning settings. An excessively high score threshold results in too many most relevant keywords, failing to effectively perturb the data, hence not significantly improving accuracy. Similarly, a too low score threshold can lead to more random perturbations. Therefore, the selection of the score threshold should be stringent.

C Efficiency and Scalability of PECOLA

We adopt Pythia (Biderman et al., 2023) as the base model of PECOLA with different scales, *i.e.*, 70M, 160M, 410M, 1B, and 1.4B. We train and do experiments on one NVIDIA A100 GPU, and the performance and time consumption are in Table 11.

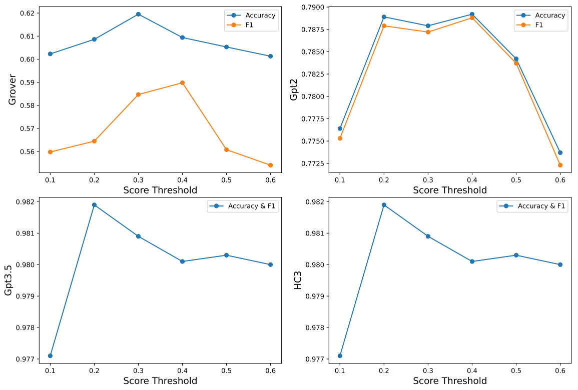

Figure 5: Effect of score threshold on model performance. In the GPT3.5 and HC3 datasets, accuracy and F1-score coincide.

With the increase in model size, both accuracy and F1-score show upward trends, while the time increase is linear, which is reasonable.

Model	**70M**	**160M**	**410M**	**1B**	**1.4B**
Acc	$58.42_{0.70}$	$63.66_{0.17}$	$71.07_{1.63}$	$72.13_{1.63}$	$74.05_{1.77}$
F1	$58.03_{0.79}$	$63.54_{0.28}$	$70.87_{1.92}$	$71.75_{2.67}$	$73.85_{1.55}$
Per epoch	16s	34s	85s	97s	113s
Single data	2.2ms	7.0ms	13.8ms	14.1ms	16.6ms

Table 11: Results of fine-tuning PECOLA with Pythia models of various scales, on the 64-example GPT2 dataset. We also demonstrate the training time per epoch and the single data test time.

D Further Clarification on Perturbation

Conversely, diversity assesses the range and variability of perturbed data, utilizing metrics Dist-1 and Dist-2 (Celikyilmaz et al., 2020). Here, we use three common perturbation methods to demonstrate the importance of not arbitrarily changing important tokens and the significance of select masks. (1) Token Substitution (Zhang et al., 2015), replaces tokens with synonyms from WordNet (Miller, 1992); (2) SwitchOut (Wang et al., 2018), uniformly samples and randomly substitutes from the vocabulary of test samples; and (3) Two-stage (Wei et al., 2021) trains the mask-filling model on the original data.

The ideal perturbation result is to have high Affinity scores while ensuring high Diversity scores (Celikyilmaz et al., 2020). As shown in Table 12, through Selective Strategy Perturbation, models achieve better diversity with high distribution shifts. And the overall improvement in Affinity by over 20% also shows greater diversity than the original data. The above results demonstrate the superiority of our perturbation method.

Method	Affinity	Diversity	
		Dist-1	Dist-2
Original	-	8.70	50.32
Token Substitution	-20.00	3.38	43.43
SwitchOut	-22.06	6.81	53.61
Two-stage	-21.13	3.24	41.85
PECOLA Mask-Filling	-1.34	15.59	57.01

Table 12: Affinity and Diversity on GPT-2 datasets.

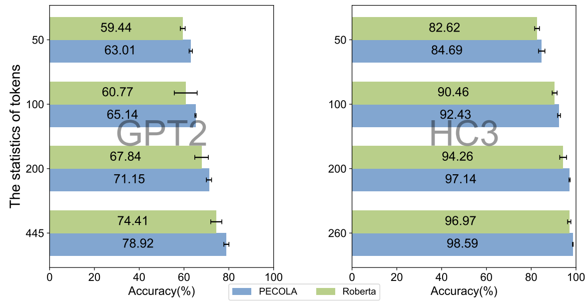

Figure 6: Performance of PECOLA and RoBERTa to detect shorter texts. The average token number of the original GPT-2 and HC3 datasets are 445 and 260.

D.1 Impact of the Chosen Mask-filling Models

This section shows the full experimental results of different mask-filling models, as shown in Table 13, the experimental results confirm the same outcomes as in the few-shot learning settings, where the T5 filling model does not perform the best across all datasets. All the above models are obtained from huggingface transformers (Wolf et al., 2020). And we do not intervene in the temperature sampling of the mask-filling model, setting it all to 1.

E Detecting Shorter Texts

To examine the efficiency of PECOLA to detect the short MGTs, we chunk the samples of GPT-2 and HC3 datasets into segments of 50, 100, and 200 tokens. As shown in Fig. 6, PECOLA consistently outperforms RoBERTa, with an average accuracy outperformance of 4.16% and 2.13% on the GPT-2 and HC3 datasets. And the relative performance decrease of PECOLA while the length shrinking is much less than RoBERTa.

Dataset	Method	Shot	*BART*	*Bert*	*GPT-2*	*Twhin Bert*	*XLM*	*XLNet*	*RoBERTa*	*T5*
Grover	*Acc*	128	$62.04_{2.51}$	$61.55_{1.74}$	$62.82_{1.24}$	$61.00_{2.20}$	$61.82_{0.82}$	$60.16_{0.43}$	$63.10_{1.76}$	$\mathbf{63.60_{1.71}}$
		512	$72.24_{1.54}$	$71.67_{1.04}$	$72.62_{1.12}$	$72.78_{1.14}$	$72.13_{0.64}$	$72.72_{1.03}$	$\mathbf{73.25_{0.84}}$	$73.12_{0.84}$
	F1	128	$57.80_{1.28}$	$57.60_{1.93}$	$58.55_{0.80}$	$56.74_{0.48}$	$57.60_{0.92}$	$56.62_{0.64}$	$58.29_{1.12}$	$\mathbf{58.98_{1.58}}$
		512	$66.25_{2.34}$	$65.56_{1.76}$	$66.72_{2.00}$	$68.49_{1.04}$	$66.38_{2.21}$	$67.50_{2.61}$	$67.49_{1.68}$	$\mathbf{68.24_{1.64}}$
	Recall	128	$58.03_{0.99}$	$57.91_{2.08}$	$\mathbf{58.72_{0.87}}$	$57.18_{0.86}$	$57.78_{1.04}$	$57.00_{0.80}$	$58.31_{0.99}$	$57.89_{1.44}$
		512	$65.85_{2.66}$	$64.87_{1.71}$	$66.01_{2.06}$	$\mathbf{68.11_{1.16}}$	$65.87_{2.46}$	$67.05_{3.04}$	$66.73_{1.68}$	$66.51_{1.64}$
GPT-2	*Acc*	128	$82.16_{1.04}$	$80.77_{0.48}$	$82.42_{1.05}$	$82.17_{0.40}$	$81.15_{0.31}$	$81.26_{0.36}$	$81.27_{1.20}$	$\mathbf{82.58_{0.49}}$
		512	$85.41_{0.66}$	$85.43_{0.53}$	$85.52_{0.57}$	$85.72_{0.39}$	$85.10_{0.27}$	$85.13_{0.60}$	$85.75_{0.55}$	$\mathbf{85.75_{0.69}}$
	F1	128	$82.12_{1.07}$	$80.67_{0.54}$	$82.38_{1.08}$	$82.12_{0.38}$	$81.11_{0.34}$	$81.24_{0.37}$	$81.16_{1.27}$	$\mathbf{82.54_{0.51}}$
		512	$85.40_{0.67}$	$85.41_{0.53}$	$85.72_{0.70}$	$85.72_{0.39}$	$85.10_{0.27}$	$85.13_{0.60}$	$\mathbf{85.75_{0.55}}$	$85.72_{0.70}$
	Recall	128	$82.15_{1.05}$	$80.75_{0.48}$	$82.01_{0.68}$	$82.17_{0.40}$	$81.15_{0.31}$	$81.26_{0.36}$	$81.25_{1.20}$	$\mathbf{82.57_{0.49}}$
		512	$85.41_{0.66}$	$85.43_{0.53}$	$\mathbf{85.80_{0.27}}$	$85.72_{0.39}$	$85.10_{0.27}$	$85.13_{0.60}$	$85.75_{0.55}$	$85.52_{0.57}$
GPT-3.5	*Acc*	128	$98.24_{0.16}$	$98.09_{0.25}$	$98.09_{0.10}$	$98.11_{0.11}$	$97.98_{0.14}$	$98.13_{0.08}$	$98.01_{0.18}$	$\mathbf{98.63_{0.32}}$
		512	$99.19_{0.13}$	$99.05_{0.15}$	$99.13_{0.17}$	$98.89_{0.21}$	$98.88_{0.21}$	$\mathbf{99.23_{0.26}}$	$99.16_{0.14}$	$99.15_{0.11}$
	F1	128	$98.24_{0.16}$	$98.09_{0.25}$	$98.09_{0.10}$	$98.11_{0.11}$	$97.98_{0.14}$	$98.13_{0.08}$	$98.01_{0.18}$	$\mathbf{98.63_{0.32}}$
		512	$99.19_{0.13}$	$99.05_{0.15}$	$99.13_{0.17}$	$98.89_{0.21}$	$98.88_{0.21}$	$\mathbf{99.23_{0.26}}$	$99.16_{0.14}$	$99.15_{0.11}$
	Recall	128	$98.24_{0.16}$	$98.09_{0.25}$	$98.09_{0.10}$	$98.11_{0.11}$	$97.98_{0.14}$	$98.13_{0.08}$	$98.01_{0.18}$	$\mathbf{98.63_{0.32}}$
		512	$99.19_{0.13}$	$99.05_{0.15}$	$99.13_{0.17}$	$98.89_{0.21}$	$98.88_{0.21}$	$\mathbf{99.23_{0.26}}$	$99.16_{0.14}$	$99.15_{0.11}$
HC3	*Acc*	128	$98.63_{0.18}$	$98.03_{0.40}$	$98.59_{0.16}$	$98.58_{0.22}$	$98.24_{0.09}$	$98.35_{0.12}$	$\mathbf{98.79_{0.32}}$	$98.06_{0.12}$
		512	$98.82_{0.35}$	$98.45_{0.21}$	$98.96_{0.25}$	$98.83_{0.24}$	$98.80_{0.38}$	$98.80_{0.30}$	$99.02_{0.23}$	$\mathbf{99.14_{0.15}}$
	F1	128	$98.63_{0.18}$	$98.03_{0.40}$	$98.59_{0.16}$	$98.58_{0.22}$	$98.24_{0.09}$	$98.35_{0.12}$	$\mathbf{98.79_{0.32}}$	$98.06_{0.12}$
		512	$98.82_{0.35}$	$98.45_{0.21}$	$98.96_{0.25}$	$98.83_{0.24}$	$98.80_{0.38}$	$98.80_{0.30}$	$99.02_{0.23}$	$\mathbf{99.14_{0.15}}$
	Reacall	128	$98.63_{0.18}$	$98.03_{0.40}$	$98.59_{0.16}$	$98.58_{0.22}$	$98.24_{0.09}$	$98.35_{0.12}$	$\mathbf{98.79_{0.32}}$	$98.63_{0.32}$
		512	$98.82_{0.35}$	$98.45_{0.21}$	$98.96_{0.25}$	$98.83_{0.24}$	$98.80_{0.38}$	$98.80_{0.30}$	$99.02_{0.23}$	$\mathbf{99.15_{0.11}}$

Table 13: The full MGT detection performance of different mask-filling models on four datasets.Comparison results between T5-large and other mask-filling models are almost model-base, except for the GPT-2 medium model.