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

Shengchao Liu¹, Xiaoming Liu¹,*, Yichen Wang¹, Zehua Cheng¹, Chengzhengxu Li¹, Zaohan Zhang², Yu Lan¹, Chao Shen¹

¹Faculty of Electronic and Information Engineering, Xi’an Jiaotong University
²Queen Mary University of London
{liusc, yichen.wang, czh2022, czx.li}@stu.xjtu.edu.cn
{xm.liu, ylan2020, chaoshen}@xjtu.edu.cn
{zhaohan.zhang@qmul.ac.uk

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, the random perturbation strategy could introduce noise, and logit regression depends on the 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 methods 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 (SOTA) 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 categories (Wu et al., 2023a; Wang et al., 2024), i.e., fine-tuned methods (Liu et al., 2023; Hu et al., 2023; Verma et al., 2023; OpenAI, 2023; Mao et al., 2024) and zero-shot metric-based methods (Gehrmann et al., 2019; Mitchell et al., 2023; Yang et al., 2023; Bao et al., 2024; Wu et al., 2023b). 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, in most cases. On the other hand, zero-shot metric-based methods show better interpretability than fine-tuned ones.

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 and uses T5 (Raffel et al., 2020) to fill in. It posits that minor perturbations of MGT tend to introduce noise, and logit regression depends on the 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 methods 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 (SOTA) by 1.20% in accuracy on average on four public datasets. And we further analyze the effectiveness, robustness, and generalization of the method.

1 Corresponding author

The code and datasets are released at https://github.com/lsc-1/PECOLA.
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 Perturbation-based Contrastive Learning 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 token removal and substitution, as shown in Fig. 1. The experiments show that the Selective Strategy Perturbation method can improve the performance of both metrics-based (i.e., DetectGPT) and model-based methods. Secondly, we propose 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 categories. 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 generators 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. Moreover, SeqXGPT (Wang et al., 2023) utilize log probability lists from white-box LLMs as features; Sniffer (Shi et al., 2024) and GPT-Who (Venkatraman et al., 2023) place more emphasis on tracing the origin of MGT. Recently, watermarking (Kirchenbauer et al., 2023) is introduced to mitigate the risk associated with unchecked MGTs by embedding imperceptible signals within text outputs during generation. In contrast to previous methods, our approach integrates data perturbation with contrastive learning, placing 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, insertion (Wei and Zou, 2019), rewriting by LLMs (Mao et al., 2024), and fine-tuning pre-trained language models (PLMs) 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. Wang et al. (2024) utilize perturbations as stress test approaches for the robustness of MGT detectors to show their loopholes. These limitations motivate us to devise data perturbation methods tailored for MGT detection. Our approach, with selective perturbation, 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 that solely relying on cross-entropy classification loss would lead 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, which joined the advantage of metric-based and model-based detection methods, 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 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 $e_i$. Tokens with scores falling below the specified threshold $\alpha$ are then incorporated into the set of important tokens $K_i$:

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

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 $s_{i\text{mask}}$. In order to relieve the information loss caused by masking perturbation, we add regularization to the vanilla 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:

$$P_i^n = \left\{\begin{array}{ll} 1, & e_i^n \notin K_i \\ 0, & e_i^n \in K_i \end{array}\right., \quad (2)$$

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

**Mask-Filling.** Additionally, we utilize 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 $\{s_{i\text{fill}}^1, \ldots, s_{i\text{fill}}^n\}$. Similar to the above, we include all filled perturbation inputs in the training set and obtain the final training set $S = \{s_1, \ldots, s_i, s_{i\text{mask}}^1, \ldots, s_{i\text{mask}}^n, s_{i\text{fill}}^1, \ldots, s_{i\text{fill}}^n\}$.

### 3.2 Token-Level Weighted Multi-Pairwise Contrastive Learning

**Importance-based Weighted 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:

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

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 feature inputs 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 $h_1^1$ from the base PLM:

$$\mathcal{L}_{\text{pos}} = \sum_{i=1}^M \frac{1}{|P_i|} \sum_{p \in P_i} \lVert (h_1^p - h_1^p) \rVert^2, \quad (6)$$
\( \mathcal{L}_{\text{neg}} = \sum_{i=1}^{M} \frac{1}{|N_t(i)|} \sum_{n \in N_t(i)} \max \left( 0, \xi - \|h_1^i - h_1^n\|^2 \right) \),

(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 \( \xi \) is the maximum \( L_2 \) distance between pairs of inputs from the same class in the batch of training data:

\[ \xi = \max_{i=1}^{M} \max_{p \in P_t(i)} \| h_1^i - h_1^p \|^2. \]

(8)

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

\[ \mathcal{L}_{\text{con}} = \frac{1}{M} (\mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{neg}}). \]

(9)

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

\[ \mathcal{L} = \mathcal{L}_{\text{ce}} + \lambda \mathcal{L}_{\text{con}}. \]

(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 PLMs as classifiers. We use RoBERTa-base (125M).

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.²

CE+SCL (Gunnel 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 PLMs, utilizing RoBERTa-base.

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.³

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

²The 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.

³For 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.
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 subspace means the standard deviation (e.g., 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. And we also conduct a more in-depth test on the entire training set in Appendix C.3.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>Shot</th>
<th>RoBERTa</th>
<th>GLTR†</th>
<th>CE+SCL</th>
<th>CE+Margin</th>
<th>IT:Clust</th>
<th>CoCo*</th>
<th>DetectGPT†</th>
<th>Fast-Detect.†</th>
<th>PECOLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grover</td>
<td>Acc</td>
<td>32</td>
<td>48.83</td>
<td>0.31</td>
<td>56.61</td>
<td>55.86</td>
<td>0.43</td>
<td>56.79</td>
<td>0.31</td>
<td>51.57</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td>56.88</td>
<td>0.03</td>
<td>57.57</td>
<td>0.63</td>
<td>58.92</td>
<td>0.37</td>
<td>46.45</td>
<td>0.20</td>
<td>58.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128</td>
<td>59.28</td>
<td>0.91</td>
<td>58.48</td>
<td>0.34</td>
<td>60.44</td>
<td>0.85</td>
<td>50.72</td>
<td>0.30</td>
<td>58.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>70.39</td>
<td>0.12</td>
<td>62.26</td>
<td>0.37</td>
<td>72.15</td>
<td>0.18</td>
<td>56.08</td>
<td>0.47</td>
<td>70.07</td>
</tr>
<tr>
<td>F1</td>
<td>Acc</td>
<td>32</td>
<td>44.13</td>
<td>0.82</td>
<td>52.77</td>
<td>0.50</td>
<td>53.21</td>
<td>0.24</td>
<td>40.79</td>
<td>0.66</td>
<td>47.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td>52.88</td>
<td>0.52</td>
<td>52.77</td>
<td>0.39</td>
<td>59.45</td>
<td>0.29</td>
<td>46.10</td>
<td>0.25</td>
<td>44.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128</td>
<td>54.69</td>
<td>0.18</td>
<td>54.47</td>
<td>0.54</td>
<td>55.72</td>
<td>0.40</td>
<td>51.37</td>
<td>0.40</td>
<td>51.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>64.49</td>
<td>0.17</td>
<td>57.11</td>
<td>0.62</td>
<td>66.25</td>
<td>0.65</td>
<td>51.80</td>
<td>0.49</td>
<td>65.15</td>
</tr>
</tbody>
</table>

The 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 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 built-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 Chat-
GPT as they use RLHF (Kirk et al., 2023).

### 4.4 Ablation Study

To illustrate the effectiveness of the PECOLA com-
ponents, we do the ablation experiments on the Selective Strategy Perturbation stage and the Con-
trastive Learning stage on the 64-example GPT-2
dataset. We also demonstrate the Scalability of
PECOLA in Appendix C.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o. mask</td>
<td>78.00±1.40</td>
<td>77.93±1.43</td>
</tr>
<tr>
<td>w/o. mask-fill</td>
<td>77.78±1.52</td>
<td>77.72±1.83</td>
</tr>
<tr>
<td>w/o. mask.CLw</td>
<td>75.80±2.22</td>
<td>75.23±2.46</td>
</tr>
<tr>
<td>w/o. mask-fill. CLw</td>
<td>75.56±1.47</td>
<td>75.10±1.73</td>
</tr>
<tr>
<td>w/o. CLw</td>
<td>76.60±1.69</td>
<td>76.22±1.65</td>
</tr>
<tr>
<td>w/o. w</td>
<td>78.02±1.16</td>
<td>77.93±1.57</td>
</tr>
<tr>
<td>PECOLA</td>
<td>78.92±1.14</td>
<td>78.88±1.17</td>
</tr>
</tbody>
</table>

Table 2: Ablation study result of PECOLA.

**Ablation on Selective Strategy Perturbation.** In
PECOLA, the data used for training primarily in-
cludes 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,
not using mask-filling texts for training.

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

As demonstrated in Table 2, in scenarios employ-
ing only the CE loss, the Selective Strategy Per-
turbation method contributes to significant perfor-
manance improvement. Moreover, the introduction
of weighing further enhances accuracy when com-
pared 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 Strat-
ey Perturbation method to evaluate the token im-
portance for the multi-pairwise contrastive learning
method. Furthermore, within the overall frame-
work, 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 fol-
lowing the application of Selective Strategy Pertur-
bation to the text.

### 4.5 Discussion and Analysis

#### 4.5.1 Model Qualities

We analyze the model qualities, including robust-
ness 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 robust-
ness 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., dele-
tion, 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 decreases merely 1.64%, underscoring
its remarkable robustness.

<table>
<thead>
<tr>
<th>Model</th>
<th>RoBERTa</th>
<th>PECOLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>Acc</td>
<td>F1</td>
</tr>
<tr>
<td>Original</td>
<td>74.41±1.41</td>
<td>73.91±1.44</td>
</tr>
<tr>
<td>Delete</td>
<td>71.77±(2.640)</td>
<td>70.42±(3.400)</td>
</tr>
<tr>
<td>Repeat</td>
<td>64.68±(9.720)</td>
<td>61.74±(12.17)</td>
</tr>
<tr>
<td>Insert</td>
<td>50.73±(23.66)</td>
<td>36.44±(37.47)</td>
</tr>
<tr>
<td>Replace</td>
<td>52.04±(22.37)</td>
<td>39.48±(34.43)</td>
</tr>
<tr>
<td>Average</td>
<td>50.81 (-14.60)</td>
<td>52.02 (-21.88)</td>
</tr>
</tbody>
</table>

Table 3: Model robustness to four perturbations.

**Analysis on Affinity.** Affinity pertains to alter-
atations in data distribution resulting from pertur-
bations, 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, follow-
ing 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-
examplv 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 effectively preserves more distinguishable features between MGTs and HWTs.

<table>
<thead>
<tr>
<th>Model</th>
<th>RoBERTa</th>
<th>PECOLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Mask DetectGPT</td>
<td>-2.64</td>
<td>-1.64</td>
</tr>
<tr>
<td>Selective Mask PECOLA</td>
<td>-0.72</td>
<td>-1.15</td>
</tr>
<tr>
<td>Mask-Filling DetectGPT</td>
<td>-4.72</td>
<td>-2.66</td>
</tr>
<tr>
<td>Mask-Filling PECOLA</td>
<td>-1.34</td>
<td>-1.34</td>
</tr>
</tbody>
</table>

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

Analysis on Diversity 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 (TS, Zhang et al. 2015), replaces tokens with synonyms from WordNet (Miller, 1992); (2) SwitchOut (SO, Wang et al. 2018), uniformly samples and randomly substitutes from the vocabulary of test samples; and (3) Two-stage (TWs, 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 5, through Selective Strategy Perturbation, models achieve better diversity with high distribution shifts. And the overall improvement in Affinity by over 18% also shows greater diversity than the original data. The above results demonstrate the superiority of our perturbation method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Affinity</th>
<th>Dist-1</th>
<th>Dist-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>-20.00</td>
<td>3.38</td>
<td>43.43</td>
</tr>
<tr>
<td>TO</td>
<td>-22.06</td>
<td>6.81</td>
<td>53.61</td>
</tr>
<tr>
<td>TWs</td>
<td>-21.13</td>
<td>3.24</td>
<td>41.85</td>
</tr>
<tr>
<td>Original</td>
<td></td>
<td>8.70</td>
<td>50.32</td>
</tr>
</tbody>
</table>

Table 5: Affinity and Diversity on GPT-2 datasets.

4.5.2 Analysis on Selective Strategies

In this section, we compare various strategies for selection in PECOLA. Beyond the PECOLA’s importance-based perturbation method and random perturbation method (DetectGPT), 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 model (Pęzik et al., 2022) during perturbation. As shown in Table 6, the experimental results of selective perturbation outperform the random perturbation method by 1.20%, 2.04%, and 2.49% in average accuracy on the 64-example GPT2 dataset. And the importance-based strategy is the highest.

<table>
<thead>
<tr>
<th>Method</th>
<th>Random</th>
<th>Prob. Rank</th>
<th>Keyword</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yake</td>
<td>76.05</td>
<td>77.35</td>
<td>78.55</td>
<td>78.92</td>
</tr>
<tr>
<td>Perplexity</td>
<td>75.53</td>
<td>76.63</td>
<td>77.11</td>
<td>77.63</td>
</tr>
</tbody>
</table>

Table 6: Different strategies for perturbation and token-level weighting, namely Random (DetectGPT), Prob. Rank (GPT2-medium), Keyword (VLT-5), Importance (PECOLA).

Further, we test the mask-filling failure ratio across the above strategies to interpret our outperformance. We find that the random strategy leads to more masking-filling failures than selective ones, which cause execution errors. Results in Table 7 indicate that selective strategy based on token importance performs the best, decreasing the failure ratio by 3.64% than random.

<table>
<thead>
<tr>
<th>Method</th>
<th>Random</th>
<th>Prob. Rank</th>
<th>Keyword</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio (%)</td>
<td>9.20</td>
<td>7.83</td>
<td>7.80</td>
<td>5.56</td>
</tr>
</tbody>
</table>

Table 7: Mask-filling failure ratio of different perturbation strategies.

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), Twbin-bert (279M; Zhang et al. 2023), XLM (279M; Alexis et al. 2020), XLNet (110M; Yang et al. 2019), RoBERTa (125M; Liu et al. 2019), and LLaMA-2 (7B; Touvron et al. 2023). As depicted in Fig. 3, the results of all mask-filling models surpass the baseline in terms of accuracy. 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 remaining full experimental results of different mask-filling models are in Appendix C.2.

Table 9: Results of cross-generator in terms of accuracy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GPT3.5→HC3</th>
<th>HC3→GPT3.5</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td>64.60196</td>
<td>62.67241</td>
<td>63.64</td>
</tr>
<tr>
<td>DetectGPT</td>
<td>77.11</td>
<td>72.66</td>
<td>74.89</td>
</tr>
<tr>
<td>PECOLA</td>
<td>78.79819</td>
<td>72.87606</td>
<td>75.83</td>
</tr>
</tbody>
</table>

4.5.5 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. 4, 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.

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 better 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.
Acknowledgements

We thank all the anonymous reviewers and the area chair for their helpful feedback, which aided us in greatly improving the paper. This work is supported by National Key R&D Program (2023YFB3107400), National Natural Science Foundation of China (62272371, 62103323, U21B2018, 62161160337, U20B2049), Initiative Postdocs Supporting Program (BX20190275, BX20200270), China Postdoctoral Science Foundation (2019M663723, 2021M692565), Fundamental Research Funds for the Central Universities under grant (xzy012024144), and Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02).

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


Xinyang Zhang, Yury Malkov, Omar Florez, Serim Park, Brian McWilliams, Jiawei Han, and Ahmed El-Kishky. 2023. Twinh-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.

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 10. We randomly select 10 different seeds for experiments, and report average test accuracy and F1-score.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Epochs</td>
<td>30</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-5</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
</tr>
<tr>
<td>Batch Size</td>
<td>16</td>
</tr>
<tr>
<td>Mask Gap</td>
<td>2</td>
</tr>
<tr>
<td>Mask Proportion</td>
<td>10%</td>
</tr>
<tr>
<td>Score threshold</td>
<td>0.4</td>
</tr>
<tr>
<td>Pre-trained model</td>
<td>RoBERTA-base</td>
</tr>
</tbody>
</table>

Table 10: Implementation details of hyperparameters.

A.2 Dataset Meta Information

We evaluate PECOLA effectiveness from domains and generators on the HC3 dataset, which primarily includes Medicine, Finance, and Computer Science domain QA, as shown in Table 11.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Medicine</th>
<th>Finance</th>
<th>Comp. Sci.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>2585</td>
<td>8436</td>
<td>1684</td>
</tr>
</tbody>
</table>

Table 11: 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. 5, 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.

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. 6, 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.
Figure 6: Effect of score threshold on model performance. In the GPT3.5 and HC3 datasets (sub-figure 3 and 4), accuracy and F1-score coincide.

C Efficiency of PECOLA

C.1 Scalability of Base Models at Different Scales

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

C.2 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.

C.3 Further Experiments on Full Datasets

To demonstrate Pecola’s superiority over the whole training set, we conduct a more in-depth test, as shown in Table 14. We train the detector using 10,000 samples from the Grover, GPT-2, and HC3 datasets, and 7,000 samples from GPT-3.5 as our training sets. Comparatively, PECOLA outperforms the second-best results in accuracy and F1-score by 0.13% and 1.56%, 0.80% and 0.83%, 0.05% and 0.05%, 0.03% and 0.03% respectively, across four datasets.
Table 12: 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>70M</th>
<th>160M</th>
<th>410M</th>
<th>1B</th>
<th>1.4B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grover</td>
<td>Acc</td>
<td>58.42</td>
<td>63.66</td>
<td>71.07</td>
<td>72.13</td>
<td>74.05</td>
</tr>
<tr>
<td></td>
<td>F1</td>
<td>58.03</td>
<td>63.54</td>
<td>70.87</td>
<td>71.75</td>
<td>73.85</td>
</tr>
<tr>
<td></td>
<td>Per epoch</td>
<td>16s</td>
<td>34s</td>
<td>85s</td>
<td>97s</td>
<td>113s</td>
</tr>
<tr>
<td></td>
<td>Single data</td>
<td>2.2ms</td>
<td>7.0ms</td>
<td>13.8ms</td>
<td>14.1ms</td>
<td>16.6ms</td>
</tr>
</tbody>
</table>

Table 13: The full MGT detection performance of different mask-filling models on four datasets. We use the model version with the same level model size, i.e. base version for most models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Shot</th>
<th>Metric</th>
<th>RoBERTa</th>
<th>GLTR</th>
<th>CE+SCL</th>
<th>CE+Margin</th>
<th>IT:Clust</th>
<th>CoCo</th>
<th>DetectGPT</th>
<th>Fast-Detect.</th>
<th>PECOLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grover</td>
<td>10000</td>
<td>Acc</td>
<td>86.13</td>
<td>60.40</td>
<td>86.57</td>
<td>86.25</td>
<td>72.65</td>
<td>85.23</td>
<td>61.42</td>
<td>65.49</td>
<td>86.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>84.07</td>
<td>59.82</td>
<td>84.95</td>
<td>85.10</td>
<td>63.23</td>
<td>83.67</td>
<td>54.28</td>
<td>63.29</td>
<td>86.66</td>
</tr>
<tr>
<td>GPT-2</td>
<td>10000</td>
<td>Acc</td>
<td>89.56</td>
<td>77.55</td>
<td>90.19</td>
<td>90.30</td>
<td>81.65</td>
<td>89.78</td>
<td>78.74</td>
<td>80.06</td>
<td>91.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>89.51</td>
<td>76.39</td>
<td>90.15</td>
<td>90.27</td>
<td>81.54</td>
<td>89.01</td>
<td>71.13</td>
<td>80.64</td>
<td>91.10</td>
</tr>
<tr>
<td>GPT-3.5</td>
<td>7000</td>
<td>Acc</td>
<td>99.89</td>
<td>93.50</td>
<td>99.74</td>
<td>99.90</td>
<td>99.09</td>
<td>99.44</td>
<td>90.80</td>
<td>94.72</td>
<td>99.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>99.89</td>
<td>93.58</td>
<td>99.74</td>
<td>99.90</td>
<td>99.09</td>
<td>99.44</td>
<td>89.14</td>
<td>94.76</td>
<td>99.95</td>
</tr>
<tr>
<td>HC3</td>
<td>10000</td>
<td>Acc</td>
<td>99.84</td>
<td>98.39</td>
<td>99.89</td>
<td>99.86</td>
<td>98.80</td>
<td>99.46</td>
<td>95.13</td>
<td>98.32</td>
<td>99.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>99.84</td>
<td>98.49</td>
<td>99.89</td>
<td>99.86</td>
<td>98.80</td>
<td>99.46</td>
<td>95.05</td>
<td>98.02</td>
<td>99.92</td>
</tr>
</tbody>
</table>

Table 14: Performance comparison of PECOLA to baseline methods on the full datasets. The results are average values of 5 runs with different random seeds. Bold shows the best and second-best results within each column.