Detecting LLM-Assisted Cheating on Open-Ended Writing Tasks on Language Proficiency Tests

Chenhao Niu and **Kevin P. Yancey** and **Ruidong Liu** and
Mirza Basim Baig and **André Kenji Horie** and **James Sharpnack**
Duolingo, Inc.
5900 Penn Ave
Pittsburgh, PA 15206
{chenhao, kyancey, ruidong, basim, andre, james.sharpnack}@duolingo.com

Abstract

The high capability of recent Large Language Models (LLMs) has led to concerns about possible misuse as cheating assistants in open-ended writing tasks in assessments. Although various detecting methods have been proposed, most of them have not been evaluated on or optimized for real-world samples from LLM-assisted cheating, where the generated text is often copy-typed imperfectly by the test-taker. In this paper, we present a framework for training LLM-generated text detectors that can effectively detect LLM-generated samples after being copy-typed. We enhance the existing transformer-based classifier training process with contrastive learning on constructed pairwise data and self-training on unlabeled data, and evaluate the improvements on a real-world dataset from the Duolingo English Test (DET), a high-stakes online English proficiency test. Our experiments demonstrate that the improved model outperforms the original transformer-based classifier and other baselines.

1 Introduction

Language proficiency tests are crucial in many high-stakes decisions, such as immigration, school admissions, and employment. To ensure valid results, testing agencies have developed security technologies to prevent and detect cheating. With the rise of Large Language Models (LLMs), concerns about LLM-assisted cheating have emerged, especially for open-ended writing questions. For example, multimodal LLMs like GPT-4 Vision (OpenAI, 2023) can provide high-quality answers from a screenshot of the question (Wu et al., 2023b).

To counteract LLM misuse, researchers have proposed methods to distinguish LLM-generated text from human-written text, which have shown good performance on domain-specific datasets (He et al., 2023; Macko et al., 2023). However, due to the difficulty of collecting real-world LLM-generated samples in malicious use cases, most research on LLM-generated text detection is conducted on datasets where positive samples are generated by researchers using LLMs (sometimes with paraphrasers). Consequently, it is unclear whether detection methods have the same performance on real-world samples where manual modifications are common. For instance, in an online test with screen recording and key tracking enabled, test-takers have to copy-type the LLM-generated text within a tight time limit, introducing typos and other modifications that rarely exist in researcher-generated samples.

In this paper, we address the detection of LLM-assisted cheating on open-ended writing tasks on English proficiency tests by bridging the gap between researcher-constructed and real-world LLM-generated samples. First, we create a dataset with human-written responses and GPT-4-generated responses, augmented with an approximated copy-typing error insertion and correction process (Section 3.2). We then fine-tune a RoBERTa-base model (Liu et al., 2019) on this dataset with the SimCLR (Chen et al., 2020) contrastive learning framework (Section 3.3). To incorporate real-world positive samples, we perform self-training (Zou et al., 2018) on pseudo-labeled samples using the initial classifier (Section 3.4). Finally, we evaluate the model on both constructed and real-world data (Section 4 and 5), which shows that the detection rate[1] is improved by 1.7x over the initial fine-tuned RoBERTa-base model at a low false positive rate of 0.1%.

2 Related Works

Our work is an application of generated text detection, enhanced with contrastive learning and self-training. We briefly review related works as follows:

[1]Detection rate: the predicted positive rate in tests with violations during the first half of the year 2024. See $\text{PPR}_{0.1\%}$ in Table 2.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 940–953
November 12-16, 2024

Generated Text Detection (GTD) is a text classification task of whether a text sample is generated by machine (or LLM specifically), or written by human. Methods of GTD fall into two categories: (1) metric-based methods that do not require model training and are usually based on the LLM-generated text distribution, such as DetectGPT (Mitchell et al., 2023), or DNA-GPT (Yang et al., 2024), and (2) model-based methods that involve training classifiers, either transformer-based (Chen et al., 2023; Hu et al., 2023) or feature-based (Wu et al., 2023a). Specifically, Yan et al. (2023) apply both transformer-based and feature-based models in detecting GPT-3-generated essays in English proficiency tests. Among these methods, fine-tuned transformer-based classifiers show high performance on benchmarks (He et al., 2023; Macko et al., 2023). However, most existing research relies on positive samples generated by researchers prompting LLMs, instead of those from malicious users. To fill this gap, our work provides an approach to evaluate and improve the classifier on real-world samples from LLM-assisted cheating.

Contrastive Learning (Hadsell et al., 2006) is a technique of learning representations by contrasting positive and negative pairs. Following SimCLR (Chen et al., 2020), a contrastive learning framework in computer vision, Pan et al. (2022) apply SimCLR to text classification to improve robustness towards adversarial samples. Bhattacharjee et al. (2023) combine SimCLR with domain-adaption to detect LLM-generated text from unseen LLMs. Inspired by these prior works, we adopt the SimCLR framework for robustness towards modifications in copy-typed LLM-generated samples.

Self-Training (Scudder, 1965) is a semi-supervised learning method where a model is initially trained on a labeled dataset, and then applied to an unlabeled dataset to obtain pseudo-labeled samples for the next round of training. It has been applied in text classification to leverage unlabeled data in low-resource settings (Sosea and Caragea, 2022; Mukherjee and Awadallah, 2020) and to use unlabeled non-English samples for cross-lingual transfer (Dong and De Melo, 2019). Similarly, we apply self-training to utilize copy-typed LLM-generated samples in unlabeled real-world data.

3 Methodology

In this section, we frame the problem of detecting copy-typed LLM-generated responses by defining probabilistic distributions for different response processes (Section 3.1). Then, we construct our main dataset by approximating the copy-typing process (Section 3.2) which enables contrastive learning (Section 3.3). We further improve the model with self-training in Section 3.4.

3.1 Problem Framing

Context. The dataset is collected from the Duolingo English Test (DET) (Cardwell et al., 2024), a high-stakes online English proficiency test. The DET employs various security measures, including video recording, screen sharing, and input monitoring. After each test session is completed and uploaded, an asynchronous proctoring process is conducted, which combines AI algorithms and human proctors to detect rule violations. In this research, we focus on an open-ended writing task in the DET, where test takers have 30 seconds to read a question given by text (see Appendix A.1 for examples), and 5 minutes to type their response on a computer. Since copy-pasting is disabled for security reasons, cheating with LLMs requires manually typing the generated responses (also known as copy-typing).

Definitions. In this context, we define the following 3 distributions to frame this problem. For notations: Let $\mathcal{S}$ be the set of all possible text sequences, $\mathbf{s} \in \mathcal{S}$ be an observed response in the given context, and $\mathbf{s}^* \in \mathcal{S}$ be an unobserved text sequence potentially related to an $\mathbf{s}$. Assume that all the following distributions are supported on $\mathcal{S}$.

1. **LLM-generated text**: Let $\mathbb{P}_{G^*}(\mathbf{s}^*)$ be the probability of an LLM generating $\mathbf{s}^*$ as a response to an input prompt asking it to complete to a writing task.
2. **Human-written text**: Let $\mathbb{P}_H(\mathbf{s})$ be the probability of $\mathbf{s}$ written by human test takers.
3. **Copy-typing process**: Let $\mathbb{P}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)$ be the probability that $\mathbf{s}$ is the result of copy-typing the given $\mathbf{s}^*$, in the context of cheating by copy-typing an LLM-generated response during the test. Intuitively, highly probable values of $\mathbf{s}$ include $\mathbf{s}^*$ and $\mathbf{s}^*$ with various errors, such as typos, misspellings, omissions, word replacements, and/or being cut off due to time limit. As most cheaters have limited English proficiency, we expect $\mathbb{P}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)$ in this context to be less concentrated at $\mathbb{P}_{\text{CT}}(\mathbf{s} = \mathbf{s}^*|\mathbf{s}^*)$, compared with copy-typing by native English speakers in a less stressful environment.

Then, based on the 3 distributions, we define the derived distributions:

1. **Copy-typed LLM-generated text**: $\mathbb{P}_G(\mathbf{s}) := \sum_{\mathbf{s}^* \in \mathcal{S}} \mathbb{P}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)\mathbb{P}_{G^*}(\mathbf{s}^*)$.
2. **Reversed copy-typing process**: $\mathbb{P}_{\text{CT}^{-1}}(\mathbf{s}^*|\mathbf{s}) := \mathbb{P}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)\mathbb{P}_{G^*}(\mathbf{s}^*)/\mathbb{P}_G(\mathbf{s})$.
3. **Reverse-copy-typed human-written text**: $\mathbb{P}_{H^*}(\mathbf{s}^*) := \sum_{\mathbf{s} \in \mathcal{S}} \mathbb{P}_{\text{CT}^{-1}}(\mathbf{s}^*|\mathbf{s})\mathbb{P}_H(\mathbf{s})$. Intuitively, this means that $\mathbf{s} \sim \mathbb{P}_H$ can be viewed as a copy-typed version of $\mathbf{s}^* \sim \mathbb{P}_{H^*}$. Note that this is a hypothetical distribution mainly for data construction, and its practical meaning is less important.
4. **Joint distributions**: $\mathbb{P}_{G,G^*}(\mathbf{s}, \mathbf{s}^*) := \mathbb{P}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)\mathbb{P}_{G^*}(\mathbf{s}^*)$, $\mathbb{P}_{H,H^*}(\mathbf{s}, \mathbf{s}^*) := \mathbb{P}_{\text{CT}^{-1}}(\mathbf{s}^*|\mathbf{s})\mathbb{P}_H(\mathbf{s})$.

Goal. With this notation, the goal of detecting LLM-assisted cheating is to determine whether a sample $\mathbf{s} \in \mathcal{S}$ is more likely to be a human-written response or a copy-typed LLM-generated one. That is, whether $\mathbb{P}_H(\mathbf{s}) > \mathbb{P}_G(\mathbf{s})$.

Challenge. A straightforward method to achieve the goal is to train a binary text classifier with labeled data from $\mathbb{P}_H$ and $\mathbb{P}_G$. However, in this case, researchers only have access to $\mathbb{P}_H$ (from honest test takers) and $\mathbb{P}_{G^*}$ (from LLMs). Neither $\mathbb{P}_G$ nor $\mathbb{P}_{\text{CT}}$ is available to researchers, as cheaters would rarely reveal whether their reference source was LLM after being caught. When the difference between $\mathbb{P}_{G^*}$ and $\mathbb{P}_G$ is large, a classifier trained on positive samples from $\mathbb{P}_{G^*}$ can be less effective in detecting samples from $\mathbb{P}_G$.

3.2 Pairwise Data Construction

To bridge the gap between $\mathbb{P}_{G^*}$ and $\mathbb{P}_G$, we construct pairwise samples $(\mathbf{s}, \mathbf{s}^*)$ by approximating the joint distributions $\mathbb{P}_{G,G^*}$ and $\mathbb{P}_{H,H^*}$, then apply contrastive learning with the pairwise samples.

Negative Samples: $(\mathbf{s}, \mathbf{s}^*) \sim \mathbb{P}_{H,H^*}$**.** We collect human-written samples $\mathbf{s} \sim \mathbb{P}_H$ from certified[2] tests before the release of ChatGPT (OpenAI, 2022) on November 30, 2022, to ensure minimal LLM-produced responses. To approximate the reverse copy-typing process $\mathbb{P}_{\text{CT}^{-1}}$, we use GPT-4 to correct errors and incompleteness in the human-written sample (see Appendix A.4 for the prompt), which produces samples $\mathbf{s}^* \sim \widehat{\mathbb{P}}_{\text{CT}^{-1}}(\cdot|\mathbf{s})$ from the approximate conditional distribution. In this way, we approximate the joint distribution $\mathbb{P}_{H,H^*}$ with $\widehat{\mathbb{P}}_{\text{CT}^{-1}}(\mathbf{s}^*|\mathbf{s})\mathbb{P}_H(\mathbf{s})$.

Positive Samples: $(\mathbf{s}, \mathbf{s}^*) \sim \mathbb{P}_{G,G^*}$**.** We prompt GPT-4 to generate samples $\mathbf{s}^* \sim \mathbb{P}_{G^*}$, with a pool of 200 prompt templates to ensure diversity of generated samples. To approximate the copy-typing process $\mathbb{P}_{\text{CT}}$, we use TextAttack (Morris et al., 2020) to insert errors into sample $\mathbf{s}^*$, denoted as $\mathbf{s} \sim \widehat{\mathbb{P}}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)$. In this way, we approximate the joint distribution $\mathbb{P}_{G,G^*}$ with $\widehat{\mathbb{P}}_{\text{CT}}(\mathbf{s}|\mathbf{s}^*)\mathbb{P}_{G^*}(\mathbf{s}^*)$. See Appendix A for more details and examples.

Dataset Splitting. To split the dataset into training/validation/test sets, we first split 912 writing questions and 200 prompt templates by 60/20/20 for training/validation/test, ensuring no overlap in writing questions or prompt templates between splits. Then, for negative samples, we randomly select at most 10 human-written samples ($\mathbf{s} \sim \mathbb{P}_H$) per question and use text correction to generate $\mathbf{s}^* \sim \widehat{\mathbb{P}}_{\text{CT}^{-1}}(\cdot|\mathbf{s})$; for positive samples, we use GPT-4 to generate the same number of samples ($\mathbf{s}^* \sim \mathbb{P}_{G^*}$) per question with randomly selected prompt templates within the set, and apply error insertion to generate $\mathbf{s} \sim \widehat{\mathbb{P}}_{\text{CT}}(\cdot|\mathbf{s}^*)$. To accurately evaluate performance at a low FPR, such as 0.1% (see Section 5), we increase the number of negative samples in the test split from 1,786 to 100,000 by collecting additional human-written responses from certified tests for the corresponding 183 writing questions. Table 1 shows the size of each split. The average number of tokens per sample is 103 for negatives and 166 for positives.[3]

Split	# Q	# Tpl	# G	# H
Training	547	120	5,338	5,338
Validation	182	40	1,776	1,776
Test	183	40	1,786	100,000

Table 1: Size of each split of the main dataset. **# Q**: number of unique writing questions. **# Tpl**: number of unique prompt templates used for generation. **# G**: number of positive samples. **# H**: number of negative samples.

[2] A test is *certified* if no violation is found during the proctoring process mentioned in Section 3.1.

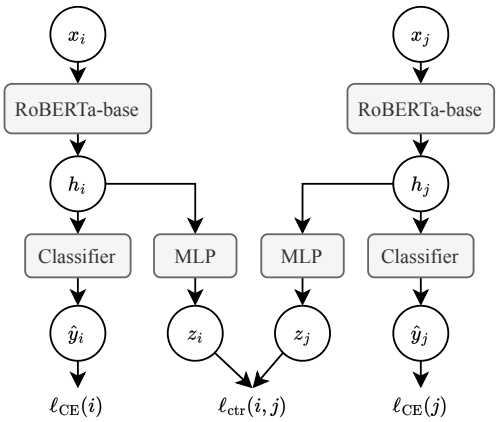

Figure 1: Network architecture for a pair of samples (x_i, x_j) in a mini-batch. There are three trainable components: (1) RoBERTa-base, the pre-trained transformer; (2) Classifier, a two-layer dense network; (3) MLP, the nonlinear projection in Section 3.3.

3.3 Contrastive Learning

Intuitively, we use contrastive learning in the fine-tuning process to guide the text embeddings to be similar for samples before and after the copy-typing process, so that the classifier can make accurate predictions regardless of copy-typing. Following the architecture used by Pan et al. (2022) and Bhattacharjee et al. (2023), we apply SimCLR (Chen et al., 2020) with RoBETRa-base (Liu et al., 2019), and train the model with pairwise data $(\mathbf{s}, \mathbf{s}^*)$. Figure 1 shows the model architecture.

Notations. For a mini-batch of N pairs of samples $\{(\mathbf{s}_k, \mathbf{s}_k^*, c_k)\}_{k=1}^N$, where $c_k \in \{0, 1\}$ is a binary label of whether $(\mathbf{s}_k, \mathbf{s}_k^*)$ are positive samples in Section 3.2. For ease of later reference, following Chen et al.'s (2020) annotations, we reindex the mini-batch as $\{(\mathbf{s}_1, c_1), (\mathbf{s}_1^*, c_1), (\mathbf{s}_2, c_2), (\mathbf{s}_2^*, c_2), \dots\} = \{(x_i, y_i)\}_{i=1}^{2N}$. That is, $x_{2k-1} = \mathbf{s}_k$, $x_{2k} = \mathbf{s}_k^*$, $y_{2k-1} = y_{2k} = c_k$. With this indexing, (x_i, x_j) is a pairwise sample $(\mathbf{s}, \mathbf{s}^*)$ if and only if (i, j) can be written as $(2k-1, 2k)$. This is useful for defining the contrastive loss.

Contrastive Loss. For the i-th sample, let $h_i \in \mathbb{R}^{d_h}$ be the final hidden state of the $[CLS]$ token of x_i in RoBERTa, and let $\hat{y}_i \in (0, 1)$ be the final output of the classifier. Following SimCLR, we use a nonlinear projection to map h_i to $z_i \in \mathbb{R}^{d_z}$, by $z_i = W_2 \text{ReLU}(W_1 h_i)$. $W_1 \in \mathbb{R}^{d_h \times d_h}$ and $W_2 \in \mathbb{R}^{d_z \times d_h}$ are trainable parameters. This nonlinear projection is used only for training, which has been shown to improve representation quality in SimCLR. The contrastive loss $\mathcal{L}_{\text{ctr}}$ is given by

$$\mathcal{L}_{\text{ctr}} = \frac{1}{2N} \sum_{k=1}^{N} \big[\ell_{\text{ctr}}(2k-1, 2k) + \ell_{\text{ctr}}(2k, 2k-1)\big],$$

$$\ell_{\text{ctr}}(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{r=1}^{2N} \mathbb{1}_{[r \neq i]} \exp(s_{i,r}/\tau)},$$

Where $s_{i,j} = \frac{z_i^T z_j}{\|z_i\| \|z_j\|}$ is the cosine similarity between the projected embedding vectors and τ is a temperature hyperparameter.

Training Objective. The training loss is a weighted sum of binary cross-entropy loss $\mathcal{L}_{CE}$ and contrastive loss $\mathcal{L}_{\text{ctr}}$, given by

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{CE} + \lambda \mathcal{L}_{\text{ctr}}$$

Where the binary cross-entropy loss $\mathcal{L}_{CE}$ is

$$\mathcal{L}_{CE} = \frac{1}{2N} \sum_{i=1}^{2N} \ell_{CE}(i),$$
$$\ell_{CE}(i) = -[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)].$$

See Appendix B for more details on training settings and hyperparameters.

3.4 Self-Training

In Section 3.2, positive pairwise data is constructed by prompting GPT-4 and inserting errors to solve the challenge of lacking real-world positive samples. To further close this gap, we apply self-training to utilize real-world positive samples from unlabeled data.

Specifically, we collect 150,000 responses from test sessions during the second half of the year 2023[4] as an unlabeled development set, assuming some of them used LLM-generated responses as external reference.

After training the model with contrastive learning, we use the model to assign pseudo-labels on the unlabeled dataset. That is, adding some unlabeled samples to the training set with an assigned

[3]The difference in length is expected, as real-world LLM-assisted responses are likely to be longer than human-written ones on average. A classifier based solely on the number of tokens is not effective in practice. See Appendix A.5.

[4]The samples used for evaluation in Section 4.1 are excluded from this development set. This period is selected because (1) there are likely to be more copy-typed LLM-generated responses than before this time, and (2) samples in 2024 are reserved for evaluation in Table 2.

label $\tilde{y} \in \{0, 1\}$, based on the output probability $\hat{y} \in (0, 1)$. Due to class imbalance, we set pseudo-labeling thresholds for positives and negatives separately, using class-balanced self-training (Zou et al., 2018).

For the t-th iteration of self-training, the pseudo-label $\tilde{y}$ is given by

$$\tilde{y} = \begin{cases} 1, & \text{if } \hat{y} \geq \theta_t^+, \\ 0, & \text{if } \hat{y} \leq \theta_t^-, \\ \text{undecided}, & \text{otherwise}, \end{cases}$$

Where θ_t^+ is the $(100 - p_t)$-th percentile of positive predictions (i.e., $\{\hat{y} | \hat{y} \geq 0.5\}$) in the development set, and θ_t^- is the p_t-th percentile of negative ones (i.e., $\{\hat{y} | \hat{y} < 0.5\}$). We set $p_t = 15 + 5t$ and iterate for $t = 1, \ldots, 4$ as the self-paced learning policy (Zou et al., 2018). Since most of the samples are pseudo-labeled as negative, we randomly down-sample negatives to the same number as positives. Pseudo-labeled samples are paired with corrected versions as in Section 3.2, and then added to training and validation sets for the next iteration.

4 Experiment Setup

4.1 Datasets and Metrics

Test Set with GPT-4-Generated Responses. We use the test split of the main dataset, as defined in Section 3.2, to evaluate the performance in detecting unmodified LLM-generated samples from human-written ones. Since practical applications of LLM-generated text detection typically require extremely low false positive rates (FPR), in addition to Area Under the ROC Curve (denoted as AUC), we also report standardized Partial Area Under the ROC Curve (pAUC) (McClish, 1989) where the FPR is in the range $[0, 10\%]$, $[0, 1\%]$, and $[0, 0.1\%]$ (denoted as $\text{pAUC}_{10\%}$, $\text{pAUC}_{1\%}$, and $\text{pAUC}_{0.1\%}$).

Unlabeled Responses from Tests with Violations. We evaluate performance on real-world positive samples using a dataset of responses from test sessions where human proctors determined that the test-taker violated the rules or cheated, such as by using external devices during the test session. The dataset is unlabeled as it is unknown whether cheaters used LLM-generated responses as their external reference. We randomly select 5,000 such samples per quarter , and calculate the Predicted Positive Rate (PPR, the proportion of positive predictions) at FPRs of 10%, 1%, and 0.1%, denoted as $\text{PPR}_{10\%}$, $\text{PPR}_{1\%}$, and $\text{PPR}_{0.1\%}$. Given the growing popularity and usability of LLMs, we expect increased involvement of LLMs as cheating tools over time since the release of ChatGPT, and therefore a higher PPR on recent responses in this dataset indicates a higher recall for real-world positive samples.

4.2 Baselines

We use the following representative baselines (see Appendix C for results on more baselines):

- **OpenAI Detector** (Solaiman et al., 2019), a RoBERTa-large model fine-tuned to detect text generated by GPT-2 (1.5B parameters).
- **GPTZero** (Tian and Cui, 2023), a commercial AI-generated text detector that predicts probabilities of a given text sample being human written, AI generated, or mixed of the two. We experimented with version `2024-07-12-base` and output `class_probabilities["AI"]`.

As an ablation study, we compare the following versions of the fine-tuned model:

- **RoBERTa$_{\text{naive}}$**: RoBERTa-base model fine-tuned on human-written samples ($\mathbb{P}_H$) and unmodified GPT-4-generated samples ($\mathbb{P}_{G^*}$). This is a straightforward fine-tuning method used in LLM-generated text detection.
- **RoBERTa$_{\text{err}}$**: RoBERTa-base model fine-tuned on human-written samples ($\mathbb{P}_H$) and error-inserted GPT-4-generated samples ($\widehat{\mathbb{P}}_G$). This means using error insertion as data augmentation on LLM-generated samples.
- **RoBERTa$_{\text{ctr}}$**: RoBERTa-base model fine-tuned with contrastive learning on pairwise data constructed in Section 3.2.
- **RoBERTa$_{\text{ctr+st}}$**: RoBERTa$_{\text{ctr}}$ improved with self-training as described in Section 3.4.

5 Results and Discussions

We present evaluation results to verify:

1. For unmodified LLM-generated samples ($\mathbb{P}_{G^*}$): whether the fine-tuned model detects them accurately with low FPR on human-written samples ($\mathbb{P}_H$) and outperforms general-purpose detectors.
2. For copy-typed LLM-generated samples ($\mathbb{P}_G$) from real-world tests: whether contrastive learning and self-training improve detection performance over naive fine-tuning.

Dataset	**Test Set with GPT-4-Generated Responses**				**Responses from Tests with Violations**		
Composition	Positives: 1,786 GPT-4-generated samples (unmodified)				Mixed: 10,000 samples in 2024H1		
	Negatives (for both datasets): 100,000 samples from certified tests prior to ChatGPT						
Metrics	**AUC**	**pAUC$_{10\%}$**	**pAUC$_{1\%}$**	**pAUC$_{0.1\%}$**	**PPR$_{10\%}$**	**PPR$_{1\%}$**	**PPR$_{0.1\%}$**
OpenAI Detector	82.52	68.03	58.14	52.27	11.20	1.25	0.14
GPTZero*	98.74	96.87	91.64	78.83	19.45	6.95	2.65
RoBERTa$_{\text{naive}}$	**100.00**$_{\pm 0.00}$	**100.00**$_{\pm 0.00}$	**99.99**$_{\pm 0.00}$	**99.86**$_{\pm 0.03}$	19.71$_{\pm 1.00}$	7.67$_{\pm 0.32}$	3.67$_{\pm 0.29}$
RoBERTa$_{\text{err}}$	**100.00**$_{\pm 0.00}$	99.98$_{\pm 0.02}$	99.84$_{\pm 0.14}$	98.88$_{\pm 0.86}$	**20.43**$_{\pm 0.51}$	8.28$_{\pm 0.44}$	4.75$_{\pm 0.27}$
RoBERTa$_{\text{ctr}}$	99.99$_{\pm 0.01}$	99.96$_{\pm 0.01}$	99.83$_{\pm 0.05}$	98.68$_{\pm 0.50}$	18.00$_{\pm 0.77}$	8.36$_{\pm 0.39}$	5.34$_{\pm 0.09}$
RoBERTa$_{\text{ctr+st}}$	99.98$_{\pm 0.01}$	99.92$_{\pm 0.04}$	99.36$_{\pm 0.31}$	94.32$_{\pm 2.95}$	19.50$_{\pm 0.82}$	**9.63**$_{\pm 0.40}$	**6.08**$_{\pm 0.20}$

Table 2: Left part: results on the test set with unmodified GPT-4-generated responses; right part: results on responses from tests with violations during the first half of the year 2024 (2024H1), as defined in Section 4.1. All values are shown as percentages, and higher values are better. The highest values are in bold. All fine-tuned RoBERTa models are repeated with 5 seeds with the mean and standard deviation reported.
*: we down-sample negative and mixed samples to 20% for GPTZero due to the cost.

5.1 Performance on Detecting Unmodified LLM-Generated Samples

The left part of Table 2 shows the performance on the test set, where positive samples are generated by GPT-4 without modification. It shows that fine-tuned RoBERTa-base models, regardless of whether using contrastive learning and self-training, perform better than general-purpose detectors for generated text. This is especially true with low FPR like 1% and 0.1%.

Note that compared with the naively fine-tuned RoBERTa-base model, contrastive learning and self-training have neutral or negative effects. This is expected since the test set is composed of positive samples generated directly by GPT-4, while contrastive learning and self-training are designed to improve performance in copy-typed versions.

We have similar observations with positive samples generated by various versions of GPT and Claude (Anthropic, 2023). See Appendix C.

5.2 Performance on Detecting Copy-Typed LLM-Generated Samples

The right part of Table 2 shows the PPR on samples from tests with violations in 2024H1. Under the assumption that there are an unknown number of copy-typed LLM-generated responses among these samples, a higher PPR at the same FPR indicates a higher recall in detecting LLM-assisted cheating. We have the following observations:

1. In-domain fine-tuning is useful, especially when a low FPR is required: When comparing GPTZero with fine-tuned RoBERTa models, we observe that although the PPR$_{10\%}$ and PPR$_{1\%}$ of GPTZero are comparable to the best results, the PPR$_{0.1\%}$ is much lower.
2. Contrastive learning (RoBERTa$_{\text{ctr}}$) is effective, outperforming both naive fine-tuning (RoBERTa$_{\text{naive}}$) and error insertion as data augmentation (RoBERTa$_{\text{err}}$).
3. Self-training can further improve performance when combined with contrastive learning.

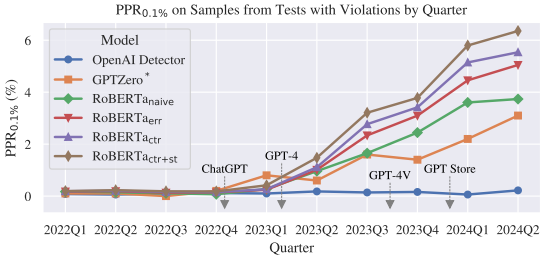

Figure 2: Predicted Positive Rate (PPR) on samples from tests with violations, with 5,000 samples per quarter. The threshold for each model is selected at FPR=0.1% on the test set. When comparing models on the same quarter, a higher PPR indicates a higher recall.

Figure 2 shows the PPR$_{0.1\%}$ on samples from tests with violations in each quarter. As supportive evidence of the increasing prevalence of LLM-assisted cheating, all fine-tuned RoBERTa models and GPTZero show a consistent upward trend in PPR$_{0.1\%}$ over time. The benefit of using contrastive learning and self-training is also consistent over time, further verifying the observation in Table 2.

Figure 3 shows how the proposed techniques change the hidden states for (predicted) copy-typed LLM-generated responses. In RoBERTa$_{\text{naive}}$, a large portion of real-world predicted positives are mapped to a separate cluster other than human-

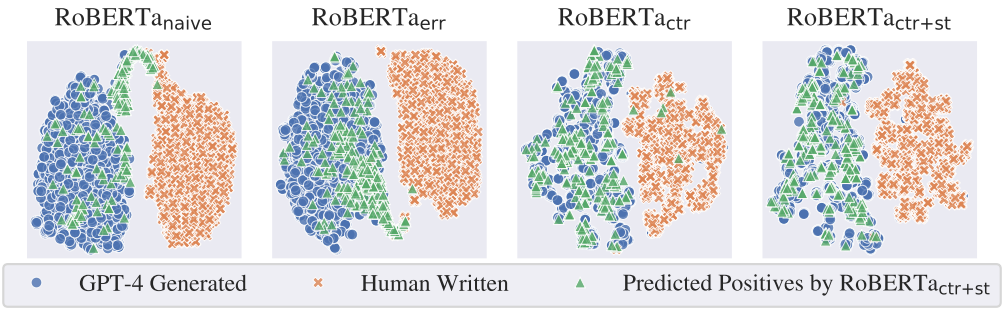

Figure 3: t-SNE (Van der Maaten and Hinton, 2008) of hidden states (h_i in Section 3.3) by the 4 versions of models, using 3 groups of samples: (1) unmodified GPT-4-generated samples in the test set, (2) human-written samples in the test set, and (3) predicted positives by RoBERTa$_{ctr+st}$ in responses from tests with violations in 2024H1.

written and GPT-generated samples. With error insertion, contrastive learning, and self-training, more predicted positive responses are mapped to the same cluster as GPT-4-generated samples, indicating better ability in detecting copy-typed LLM-generated responses.

6 Conclusion

In this work, we present a framework for training an LLM-generated text detector to effectively detect generated samples with human modification during copy-typing. We enhance the existing method of fine-tuning transformer-based classifiers by incorporating contrastive learning and self-training, and verify these improvements in detecting LLM-assisted cheating on open-ended writing tasks in an English proficiency test. This research provides new possibilities for detecting LLM misuse with post-generation modifications, such as errors introduced during copy-typing.

For future directions, we plan to further investigate characteristics of LLM-assisted cheating, such as modeling the copy-typing process. We also plan to apply the framework to other domains, such as open-ended speaking questions, where the process of reading aloud followed by automatic speech recognition can be viewed as a more complex post-generation modification than copy-typing.

Ethical Considerations

Assumption of False Positive Rate (FPR). Our analysis computes false positive rates among test-taker responses on data prior to November 30, 2022, the release of ChatGPT. Some of our analyses assume that this FPR is constant over time. However, it is possible that FPR will change over time due to linguistic drift, especially if LLMs begin to influence how test takers write and speak. While this requires further study, we think that it is unlikely that there has been enough drift to impact the results of this study in a significant way.

Inferring Cheating When LLM Responses Are Detected. Relatedly, even when a response is correctly predicted to be LLM-generated, it does not always imply cheating, depending on the rules of the language test. For example, test takers might prepare for a language test by memorizing phrases or templates from LLM-generated responses that they then use during their test. These issues should be considered when constructing policies around how these types of models should be used in a proctoring process.

Potential Applications in Test Proctoring. As a predicted positive does not always imply cheating, we caution against invalidating test sessions solely on the basis of a positive prediction of these models. Instead, other signals need to be considered by human proctors in order to minimize the risk of falsely accusing test takers of cheating. If proctors apply additional scrutiny to tests flagged by LLM-generated text detectors, confirmation bias should be evaluated and minimized as well.

Acknowledgments

We would like to express our gratitude to all those who helped review this paper and to those who provided valuable input on this research, especially William C. M. Belzak, Manqian Liao, J.R. Lockwood, Phoebe Mulcaire, Andrew Runge, Xiyan Shao and Yong-Siang Shih.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Anthropic. 2023. Claude.

Amrita Bhattacharjee, Tharindu Kumarage, Raha Moraffah, and Huan Liu. 2023. ConDA: Contrastive domain adaptation for AI-generated text detection. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 598–610, Nusa Dua, Bali. Association for Computational Linguistics.

Ramsey Cardwell, Ben Naismith, Geoffrey T. LaFlair, and Steven Nydick. 2024. Duolingo english test: technical manual. *Duolingo Research Report*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

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

Xin Luna Dong and Gerard De Melo. 2019. A robust self-learning framework for cross-lingual text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6306–6310.

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*.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text. *Preprint*, arXiv:2401.12070.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. Mgtbench: Benchmarking machine-generated text detection. *arXiv preprint arXiv:2303.14822*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *Advances in Neural Information Processing Systems*.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Dominik Macko, Robert Moro, Adaku Uchendu, Jason Lucas, Michiharu Yamashita, Matúš Pikuliak, Ivan Srba, Thai Le, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2023. MULTITuDE: Large-scale multilingual machine-generated text detection benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9960–9987, Singapore. Association for Computational Linguistics.

Donna Katzman McClish. 1989. Analyzing a portion of the roc curve. *Medical decision making*, 9(3):190–195.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962. PMLR.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Subhabrata Mukherjee and Ahmed Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212.

OpenAI. 2022. Chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Lin Pan, Chung-Wei Hang, Avirup Sil, and Saloni Potdar. 2022. Improved text classification via contrastive adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11130–11138.

Henry Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

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

Tiberiu Sosea and Cornelia Caragea. 2022. Leveraging training dynamics and self-training for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4750–4762, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Edward Tian and Alexander Cui. 2023. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023a. LLMDet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2113–2133, Singapore. Association for Computational Linguistics.

Yang Wu, Shilong Wang, Hao Yang, Tian Zheng, Hongbo Zhang, Yanyan Zhao, and Bing Qin. 2023b. An early evaluation of gpt-4v (ision). *arXiv preprint arXiv:2310.16534*.

Duanli Yan, Michael Fauss, Jiangang Hao, and Wenju Cui. 2023. Detection of ai-generated essays in writing assessment. *Psychological Testing and Assessment Modeling*, 65(2):125–144.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Ruth Petzold, William Yang Wang, and Haifeng Chen. 2024. DNA-GPT: Divergent n-gram analysis for training-free detection of GPT-generated text. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Yang Zou, Zhiding Yu, B.V.K. Vijaya Kumar, and Jinsong Wang. 2018. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

A Dataset Details

A.1 Examples of Writing Questions

1. `What is an important life lesson you have learned in the last few years? Explain why it was such an important lesson to learn.`
2. `If you could go back and redo one day in your life, what day would you choose? Explain the reasons for your choice.`

A.2 Examples of Prompt templates

As mentioned in 3.2, we use GPT-4 to generate a pool of 200 prompt templates, with 100 simple templates and 100 complex ones. Below are examples of prompt templates, where `{topic}` will be replaced by the writing question.

Simple prompt template example: `Write an essay of the topic below. Ensure your response does not exceed 120 words: {topic}.`

Complex prompt template example:
```
In the upcoming English test, you are tasked with writing an insightful essay on the given topic:
{topic}
Please ensure that your essay meets the following requirements:
1. It should be concise and manageable within a five-minute timeframe. A maximum word count of 120 words is recommended.
2. Aim to secure a high score by satisfying the following guidelines provided by the test conductors:
- Maintain linguistic accuracy by adhering to correct grammatical norms, punctuation rules, and capitalization. Both American and British spellings are acceptable.
- Enhance the readability of your essay by varying sentence structures and word choices.
A diverse and sophisticated use of vocabulary and grammar could help you fetch a higher score. Please incorporate your personal experiences, observations, and relevant examples to support your views on the topic. Be succinct, yet informative.
```

A.3 Example of GPT-4-Generated Samples

Before error insertion: `The past few years, amidst my fervent pursuits in Molecular Biology and tuneful mindsets on piano keys, I've learned a significant life lesson: balance. While engrossing myself`

in the scientific mysteries of life, I discovered that total focus can lead to burnout - hampering my performance and detaching me from my much-cherished piano sessions. Realizing that my love for music provided vital relaxation, promoting my scientific understandings like calm counterpoint amidst intricate fugue. This lesson changed my perspective on success, shifting it from all-out dedication to harmonious equilibrium, reemphasizing the importance of personal well-being in any worthwhile endeavor.

After error insertion: (Note: insertions and replacements are underlined like this, and deletions are crossed like ~~this~~. See Appendix A.6 for settings of error insertion.) BThe past few years, amidst ny fervent pursuits in Molecular Biology and tuneful mindsets on piano keys, I've learned a significant life lesson: balance. While engrossing myself in the scientific mysteries of life, I discovered that total focus can lead to burnout - hampering my performance and detaching me from my much-cherished piano sessions. Realizing that my love for music provided vital relaxation, promoting my scieqntific understandings like calm counterpoint amidst intricate fugue. This lesson changed my perspective on success, shifting it from all-out dedication jto harmonious equilibrium, reemphasizing the ~~importance of personal well-being in any worthwhile endeavor.~~

A.4 Prompt Template for Text Correction

Here is the prompt template for GPT-4 to correct human-written responses in Section 3.2. `{essay}` below will be replaced by the human-written response.

```
  Correct these mistakes in the following essay:
1. Typos and misspelling.
2.  Missing or extra punctuation or spaces.
3. Unfinished sentences.
4. Grammar mistakes.
Do not make other changes to the essay.

Return the corrected essay only, without any extra words before or after it.

The essay:

{essay}
```

A.5 Effects of Response Length

As mentioned in Section 3.2, in the main dataset, the average number of tokens per sample is 103 for human-written samples and 166 for GPT-4-generated samples. While we expect LLM-assisted responses to be longer than human-written ones in practice, this may raise concerns about how this affects the evaluation results. Here we evaluate a naive logistic regression on the number of tokens in the same way as in Table 2. On the test set with GPT-4-generated samples, although the AUC is non-trivial, the pAUCs at FPR as 1% and 0.1% are close to random guess, due to the existence of long human-written samples. In terms of unlabeled responses from tests with violations, all three PPRs are close to random guess.

- Test Set with GPT-4-Generated Responses (the left part in Table 2):
  - AUC: 86.59%
  - $\text{pAUC}_{10\%}$: 63.32%
  - $\text{pAUC}_{1\%}$: 53.89%
  - $\text{pAUC}_{0.1\%}$: 51.34%
- Response from Tests with Violations in 2024H1 (the right part in Table 2):
  - $\text{PPR}_{10\%}$: 11.67%
  - $\text{PPR}_{1\%}$: 1.14%
  - $\text{PPR}_{0.1\%}$: 0.11%

A.6 Settings of Error Insertion

We use the following settings in TextAttack (Morris et al., 2020) for error insertion in Section 3.2.

- Each sample has an 80% of chance to be modified with TextAttack, using the following methods in `textattack.transformations`, with a random number of modifications:
  - `WordSwapRandomCharacterInsertion`
  - `WordSwapRandomCharacterDeletion`
  - `WordSwapQWERTY`
- Independent from whether TextAttack is applied, each sample has a 20% chance of dropping the last n words, where n is randomly selected from $\{1, \ldots, 10\}$.

B Model Training Details

We use the following settings for model training:

- Pre-trained weights for RoBERTa-base: FacebookAI/roberta-base (Liu et al., 2019).
- Max number of tokens: 256 .
- Size of hidden state: $d_h = 768$ (the size of hidden state in RoBERTa-base).
- Size of projected hidden state: $d_z = 300$ (following Pan et al. (2022)).
- Temperature in contrastive loss: $\tau = 0.5$.
- Weight of contrastive loss in the training loss function: $\lambda = 0.5$.
- Optimizer: AdamW (Loshchilov and Hutter, 2019).
- Mini-batch size: $N = 16$.
- Learning rate: initial value is 1e-5, with `ReduceLROnPlateau` using loss on validation set.
- Max epochs: 35, with early stopping using loss on validation set.

The model is trained with 4 NVIDIA T4 GPU cards with 4-way data parallelism (i.e., each batch contains 4 mini-batches).

C Additional Results on Detecting Unmodified LLM-Generated Samples

Since the focus of this work is mainly on detecting copy-typed LLM-generated responses, rather than unmodified generated responses from various LLMs, we only include the best two baselines on the test set in Table 2, OpenAI Detector (Solaiman et al., 2019) and GPTZero (Tian and Cui, 2023). We report the results for the rest of them in this section.

C.1 Additional Baselines

We report results from the following additional baselines.

- **RADAR** (Hu et al., 2023) is a fine-tuned RoBERTa-large model with adversarial learning, for robust AI-generated text detection. We used the model checkpoint TrustSafeAI/RADAR-Vicuna-7B, which was trained with samples generated by Vicuna-7B-v1.1 (Zheng et al., 2023).
- **ChatGPT Detector** (Guo et al., 2023) is a fine-tuned RoBERTa-base model on HC3 dataset, where positive samples were generated by GPT-3.5. We used the checkpoint Hello-SimpleAI/chatgpt-detector-roberta.
- **Binocular** (Hans et al., 2024) is a zero-shot detector, based on contrasting two related language models with *cross-perplexity*. Following the original paper, we used Falcon7B and Falcon-7B-Instruct (Almazrouei et al., 2023) models to compute the Binocular score.

C.2 Evaluation Results on Generated Samples by Various LLMs

In this section we share the result on test sets, with the same 100,000 samples from certified tests as negative samples, and 1,800 positive samples generated by different versions of ChatGPT and Claude. Note that all fine-tuned RoBERTa models are the same trained instances used in Table 2. GPTZero is evaluated only on GPT-4-generated samples in Table 2 due to cost.

Table 3 to Table 9 show the results on positive samples generated by 3 versions of ChatGPT and 4 versions of Claude.

Observations:

1. The performance of baseline detectors is sensitive to the LLM used to generate positive samples. For instance, Binocular is the best performing baseline in all the experiments here, except for a worse performance than OpenAI Detector on GPT-4. This aligns with the results in the original paper for Binocular (Hans et al., 2024), where the recall on GPT-4-generated samples is lower than those from GPT-3.5-turbo. However, a comprehensive evaluation and analysis on this observation is not the focus of this work.
2. Compared to $\text{RoBERTa}_{\text{naive}}$, error insertion, contrastive learning, and self-training all have neutral or negative effects on unmodified LLM-generated samples, aligning with the observation in Table 2.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	41.23	49.16	49.96	49.97
RADAR	31.85	47.37	49.75	49.97
ChatGPT Detector	89.31	71.19	56.59	51.23
Binocular	98.83	95.97	87.94	76.15
RoBERTa$_{\text{naive}}$	$\mathbf{99.99}_{\pm 0.00}$	$\mathbf{99.92}_{\pm 0.01}$	$\mathbf{99.34}_{\pm 0.10}$	$\mathbf{97.48}_{\pm 0.14}$
RoBERTa$_{\text{err}}$	$99.94_{\pm 0.04}$	$99.67_{\pm 0.19}$	$97.86_{\pm 1.02}$	$93.28_{\pm 2.83}$
RoBERTa$_{\text{ctr}}$	$99.19_{\pm 0.41}$	$98.77_{\pm 0.43}$	$97.63_{\pm 0.16}$	$93.13_{\pm 0.89}$
RoBERTa$_{\text{ctr+st}}$	$99.40_{\pm 0.12}$	$98.86_{\pm 0.10}$	$97.50_{\pm 0.37}$	$89.23_{\pm 3.18}$

Table 3: Result on the test set with positive samples generated by GPT-3.5-turbo

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	82.25	67.87	57.97	52.20
RADAR	68.35	49.21	49.76	49.97
ChatGPT Detector	52.14	48.74	49.81	49.97
Binocular	74.10	59.55	52.30	50.34
RoBERTa$_{\text{naive}}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{99.99}_{\pm 0.00}$	$\mathbf{99.85}_{\pm 0.03}$
RoBERTa$_{\text{err}}$	$\mathbf{100.00}_{\pm 0.00}$	$99.98_{\pm 0.02}$	$99.84_{\pm 0.15}$	$98.89_{\pm 0.88}$
RoBERTa$_{\text{ctr}}$	$99.99_{\pm 0.01}$	$99.96_{\pm 0.01}$	$99.83_{\pm 0.05}$	$98.68_{\pm 0.51}$
RoBERTa$_{\text{ctr+st}}$	$99.98_{\pm 0.01}$	$99.92_{\pm 0.04}$	$99.36_{\pm 0.31}$	$94.31_{\pm 2.95}$

Table 4: Result on the test set with positive samples generated by GPT-4.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	66.58	54.89	51.41	50.19
RADAR	49.43	47.44	49.75	49.97
ChatGPT Detector	74.25	55.17	50.68	49.99
Binocular	94.63	85.17	69.55	57.84
RoBERTa$_{\text{naive}}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{99.95}_{\pm 0.02}$	$\mathbf{99.74}_{\pm 0.07}$
RoBERTa$_{\text{err}}$	$99.99_{\pm 0.00}$	$99.97_{\pm 0.02}$	$99.80_{\pm 0.16}$	$98.96_{\pm 0.90}$
RoBERTa$_{\text{ctr}}$	$99.99_{\pm 0.00}$	$99.97_{\pm 0.01}$	$99.78_{\pm 0.06}$	$98.44_{\pm 0.62}$
RoBERTa$_{\text{ctr+st}}$	$99.98_{\pm 0.00}$	$99.92_{\pm 0.03}$	$99.29_{\pm 0.32}$	$93.78_{\pm 3.07}$

Table 5: Result on the test set with positive samples generated by GPT-4o.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	35.47	48.43	49.93	49.99
RADAR	31.70	47.37	49.75	49.97
ChatGPT Detector	90.92	74.04	58.45	52.17
Binocular	99.73	98.95	95.12	85.36
RoBERTa$_{\text{naive}}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{99.98}_{\pm 0.00}$	$\mathbf{99.79}_{\pm 0.03}$	$\mathbf{98.85}_{\pm 0.11}$
RoBERTa$_{\text{err}}$	$99.97_{\pm 0.03}$	$99.82_{\pm 0.16}$	$98.75_{\pm 1.06}$	$95.02_{\pm 2.96}$
RoBERTa$_{\text{ctr}}$	$99.83_{\pm 0.09}$	$99.54_{\pm 0.22}$	$98.72_{\pm 0.20}$	$94.56_{\pm 0.89}$
RoBERTa$_{\text{ctr+st}}$	$99.87_{\pm 0.03}$	$99.64_{\pm 0.06}$	$98.31_{\pm 0.38}$	$89.90_{\pm 3.35}$

Table 6: Result on the test set with positive samples generated by Claude-3 Haiku.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	44.83	49.54	49.84	49.97
RADAR	40.52	47.45	49.76	49.97
ChatGPT Detector	93.06	77.09	59.96	52.66
Binocular	99.14	97.28	91.46	80.95
RoBERTa$_{\text{naive}}$	$\mathbf{99.99}_{\pm 0.00}$	$\mathbf{99.96}_{\pm 0.01}$	$\mathbf{99.76}_{\pm 0.04}$	$\mathbf{98.69}_{\pm 0.11}$
RoBERTa$_{\text{err}}$	$99.96_{\pm 0.03}$	$99.80_{\pm 0.18}$	$98.69_{\pm 1.12}$	$94.53_{\pm 3.49}$
RoBERTa$_{\text{ctr}}$	$99.79_{\pm 0.08}$	$99.53_{\pm 0.15}$	$98.51_{\pm 0.20}$	$93.72_{\pm 0.99}$
RoBERTa$_{\text{ctr+st}}$	$99.81_{\pm 0.05}$	$99.53_{\pm 0.07}$	$97.86_{\pm 0.47}$	$87.68_{\pm 3.75}$

Table 7: Result on the test set with positive samples generated by Claude-3 Opus.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	60.02	51.49	50.17	50.06
RADAR	38.88	47.41	49.75	49.97
ChatGPT Detector	78.54	60.08	52.50	50.38
Binocular	92.62	85.89	74.37	64.00
RoBERTa$_{\text{naive}}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{99.99}_{\pm 0.00}$	$\mathbf{99.95}_{\pm 0.01}$	$\mathbf{99.61}_{\pm 0.08}$
RoBERTa$_{\text{err}}$	$99.99_{\pm 0.01}$	$99.96_{\pm 0.05}$	$99.73_{\pm 0.23}$	$98.56_{\pm 0.94}$
RoBERTa$_{\text{ctr}}$	$99.96_{\pm 0.04}$	$99.88_{\pm 0.06}$	$99.59_{\pm 0.03}$	$98.03_{\pm 0.54}$
RoBERTa$_{\text{ctr+st}}$	$99.97_{\pm 0.01}$	$99.87_{\pm 0.04}$	$99.21_{\pm 0.31}$	$93.63_{\pm 2.90}$

Table 8: Result on the test set with positive samples generated by Claude-3 Sonnet.

Models	**AUC**	**pAUC**$_{10\%}$	**pAUC**$_{1\%}$	**pAUC**$_{0.1\%}$
OpenAI Detector	60.28	52.27	50.29	50.03
RADAR	49.70	47.47	49.75	49.97
ChatGPT Detector	85.88	64.68	53.27	50.50
Binocular	99.11	97.00	88.52	73.22
RoBERTa$_{\text{naive}}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{100.00}_{\pm 0.00}$	$\mathbf{99.96}_{\pm 0.01}$	$\mathbf{99.74}_{\pm 0.05}$
RoBERTa$_{\text{err}}$	$99.99_{\pm 0.01}$	$99.97_{\pm 0.03}$	$99.77_{\pm 0.22}$	$98.80_{\pm 1.10}$
RoBERTa$_{\text{ctr}}$	$99.96_{\pm 0.01}$	$99.92_{\pm 0.01}$	$99.74_{\pm 0.07}$	$98.33_{\pm 0.59}$
RoBERTa$_{\text{ctr+st}}$	$99.96_{\pm 0.02}$	$99.86_{\pm 0.04}$	$99.21_{\pm 0.33}$	$93.56_{\pm 3.05}$

Table 9: Result on the test set with positive samples generated by Claude-3.5 Sonnet.

D Additional Results on Detecting Copy-Typed LLM-Generated Samples

D.1 Predicted Positive Rates at Different FPR

Figure 4 shows the predicted positive rates (PPR, the proportion of positive predictions) at the threshold that the false positive rate (FPR) on the test set is 0.1%, 1%, and 10%. Similar to the observations in Table 2 and Figure 2 in Section 5, the benefit of in-domain fine-tuning (compared to GPTZero), contrastive learning, and self-training (compared to RoBERTa$_{\text{naive}}$) is more observable when a low FPR such as 0.1% is selected, and the increase in PPR is mostly consistent over time.

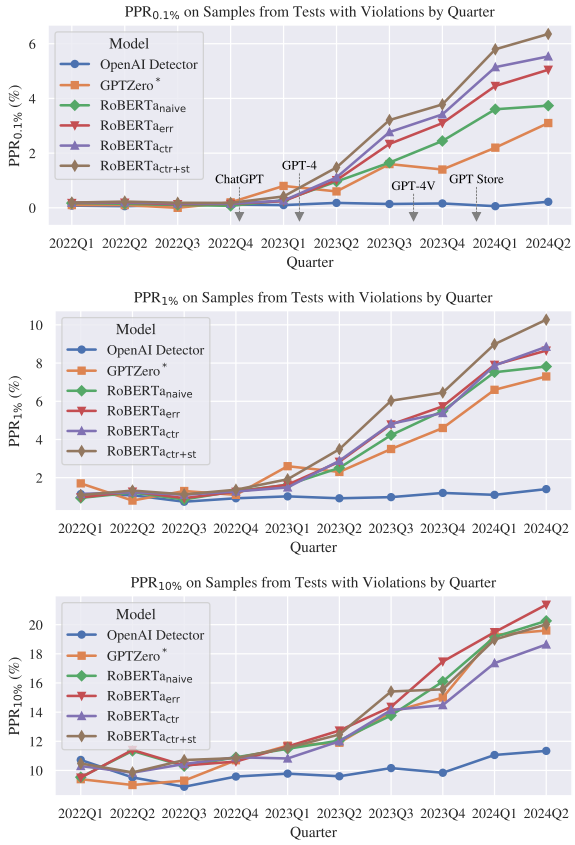

Figure 4: Predicted Positive Rate (PPR) on samples from tests with violations, with 5,000 samples per quarter. Similar to Figure 2, the threshold for each model is selected at a fixed FPR on the test set.