Iron Sharpens Iron: Defending Against Attacks in Machine-Generated Text Detection with Adversarial Training

Yuanfan Li[1,†], Zhaohan Zhang[2,†], Chengzhengxu Li[1,†], Chao Shen[1], Xiaoming Liu[1,*]

[1]Faculty of Electronic and Information Engineering, Xi'an Jiaotong University
[2]Queen Mary University of London
[†] Equal contribution, [*] Corresponding author
`liyuan7716@gmail.com`, `czx.li@stu.xjtu.edu.cn`
`zhaohan.zhang@qmul.ac.uk`, `{chaoshen, xm.liu}@xjtu.edu.cn`

Abstract

Machine-generated Text (MGT) detection is crucial for regulating and attributing online texts. While the existing MGT detectors achieve strong performance, they remain vulnerable to simple perturbations and adversarial attacks. To build an effective defense against malicious perturbations, we view MGT detection from a threat modeling perspective, that is, analyzing the model's vulnerability from an adversary's point of view and exploring effective mitigations. To this end, we introduce an adversarial framework for training a robust MGT detector, named **GRE**edy **A**dversary Promo**T**ed Defend**ER** (GREATER). The GREATER consists of two key components: an adversary GREATER-A and a detector GREATER-D. The GREATER-D learns to defend against the adversarial attack from GREATER-A and generalizes the defense to other attacks. GREATER-A identifies and perturbs the critical tokens in embedding space, along with greedy search and pruning to generate stealthy and disruptive adversarial examples. Besides, we update the GREATER-A and GREATER-D synchronously, encouraging the GREATER-D to generalize its defense to different attacks and varying attack intensities. Our experimental results across 10 text perturbation strategies and 6 adversarial attacks show that our GREATER-D reduces the Attack Success Rate (ASR) by **0.67%** compared with SOTA defense methods while our GREATER-A is demonstrated to be more effective and efficient than SOTA attack approaches. Codes and dataset are available in `https://github.com/Liyuuuuu111/GREATER`.

1 Introduction

The rapid development of large language models (LLM) (Achiam et al., 2023; Dubey et al., 2024; Anthropic, 2024; Guo et al., 2025) enables the model to generate highly human-like texts, which has raised broad concerns about the unrestricted dissemination of non-attributed textual contents in-
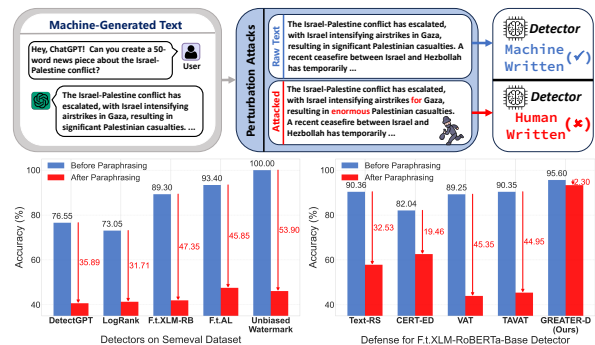

Figure 1: Performance drop of different MGT detectors and defense methods under text perturbation[1].

cluding misinformation, fabricate news, and phishing emails. These negative impacts of MGTs lead to extensive works on MGT detection (Mitchell et al., 2023; Verma et al., 2024; Liu et al., 2024b; Bao et al.; Liu et al., 2024b) to accurately attribute the authorship of textual content and inform the readers.

Despite the superb performance of current MGT detectors, a recent study (Wang et al., 2024a) finds an astonishing fact that *all* detectors exhibit different loopholes in robustness, that is, existing detectors suffer great performance drop when facing different **text perturbation strategies** including editing (Kukich, 1992; Gabrilovich and Gontmakher, 2002), paraphrasing (Shi et al., 2024), prompting (Zamfirescu-Pereira et al., 2023), and co-generating (Kushnareva et al., 2024), etc. As illustrated in Figure 1, the detection accuracy of current detectors drops by around 30%-50% when confronted with simple perturbations, and the defense methods for general text classification cannot be simply adapted to the MGT detection scenario. More se-

---

[1]The detectors include DetectGPT (Mitchell et al., 2023), LogRank (Su et al., 2023), fine-tuned xlm-roberta-base (Liu et al., 2019), fine-tuned Albert-large (Lan et al., 2019), and Unbiased Watermark (Hu et al., 2023b). The defense methods contain Text-RS (Zhang et al., 2024b), CERT-ED (Huang et al., 2024), VAT (Miyato et al., 2016), TAVAT (Li et al., 2021), and GREATER-D (ours).

riously, the vulnerability of MGT detectors is also unveiled by **adversarial attacks** that exploit the internal state (Yoo and Qi, 2021) or outputs (Liu et al., 2024a; Yu et al., 2024; Hu et al., 2024) of the detectors through multiple queries. Alas, there are few works on improving the robustness against adversarial attacks for MGT detectors.

Motivation. We rely on *threat modeling* to advance the robustness of the MGT detectors against perturbation and adversarial attacks. As the proverb says 'Iron sharpens iron', we focus on constructing powerful adversarial examples which mislead the prediction of the detector to facilitate the post-training of MGT detectors and defend against different attacks. Existing text perturbation strategies (Wang et al., 2024a) adjust token distribution without accessing information from the target MGT detector, resulting in low-quality and non-targeted adversarial examples. The adversarial attacks are only effective in white-box setting (Yoo and Qi, 2021) or require excessive queries to target detectors (Hu et al., 2024; Yu et al., 2024; Liu et al., 2024a). Moreover, Wang et al. (2024c) find that the defense built by adversarial training cannot generalize well to the attacks on which it was not originally trained. To overcome these limitations, we propose an efficient adversarial training framework that works in a black-box setting and builds generalizable defense against a wide variety of perturbations and attacks for MGT detectors.

Our Work. In this paper, we propose an adversarial framework for training robust MGT detector, namely **GRE**edy **A**dversary Promo**T**ed Defende**R** (GREATER). GREATER consists of an adversary (GREATER-A) and a detector (GREATER-D). The GREATER-D learns to discern MGTs from the human-written texts (HWTs), while the GREATER-A, which queries the output of the detector, aims to imply minimum perturbation on MGTs to deceive the detector. Restricted by the scenario where **only outputs** from the target detector are available, we use an open-sourced surrogate model to retrieve gradient information to identify important tokens in the prediction. Afterwards, we introduce a gradient ascent perturbation on the embedding of MGTs from the surrogate model to enhance both the quality and stealthiness of generated adversarial text. To reduce the number of queries needed for building effective adversarial examples, we design a greedy search and pruning strategy. In the training stage, we update the GREATER-A and GREATER-D in the same training step so that GREATER-D

learns from a curriculum of adversarial examples to generalize its defense. The experiment results demonstrate that our method achieves an average ASR of 5.53% against various attacks, which is 0.67% lower compared to the SOTA defense methods. We also find that GREATER-A achieves the most effective attack, achieving an ASR of 96.58%, which surpasses SOTA attack methods by 8.45% while requiring 4 times fewer queries.

Our contributions are as follows:

- **Adversarial Training Framework.** We propose an adversarial training framework GREATER to improve the robustness of MGT detectors, in which the adversary maliciously perturbs the MGTs to construct hard adversarial examples, while the detector is trained to maintain correct prediction towards the adversarial examples. We update the detector and the adversary generator in the same training step for better generalization on defense.

- **Adversarial Examples Generation.** We propose a strong and efficient adversarial examples construction method in the black-box setting. We retrieve gradient information from a surrogate model to rank the important tokens in MGT detection and design a greedy search and pruning strategy to reduce the query times needed for adversarial attacks.

- **Outstanding Performance.** Testing results across 16 attack methods demonstrate that our detector outperforms 10 existing SOTA defense methods in robustness, while our adversary achieves significant improvements in both attack efficiency and effectiveness compared to 13 SOTA attack approaches.

2 Related Work

Machine-Generated Text (MGT) Detection. There have been attempts to detect and attribute the MGTs in the pre-LLM era (Zhong et al., 2020; Uchendu et al., 2020). Nowadays, many works (Mitchell et al., 2023; Wang et al., 2023; Liu et al., 2022; Kushnareva et al., 2024; Guo et al.) aim to accurately annotate online texts as LLMs' astonishing ability to generate fluent, logical, and human-like content, which helps the proliferation of unchecked information. Despite the achievements made in MGT detection, some works indicate the MGT detectors are vulnerable to simple perturbation or adversarial attacks. For example,

Wang et al. (2024a) test the robustness of eight MGT detectors with twelve perturbation strategies and they surprisingly find that none of the existing detectors remain robust under all the attacks. Moreover, MGT detectors' defense against adversarial attacks is also questioned (Fishchuk, 2023). Other studies also reveal the fact that MGT detectors suffer from authorship obfuscation (Macko et al., 2024) and biased decision (Liang et al., 2023). To mitigate the vulnerability of MGT detectors, our work focuses on improving detector robustness against text perturbations and adversarial attacks.

Adversarial Training. Adversarial training aims to optimize the model toward maintaining correct predictions against adversarial examples that are misleading data constructed for malicious purposes. Earlier works first augment the training set with adversarial examples for defense against specific attacks (Huang et al., 2024; Zeng et al., 2023). These methods are shown to be hard to generalize to unseen attacks (Wang et al., 2024c). Yoo and Qi (2021) update the adversary and the target model in the same step to generalize the defense to unseen attacks. However, they rely on the availability of explicit first-order gradients, which is not applicable in the real-world case. OUTFOX (Koike et al., 2024) identifies adversarial attack with in-context learning, which requires the demonstrations of adversarial samples as prompts. RADAR (Hu et al., 2023a) uses a paraphraser as the adversary. Both OUTFOX and RADAR are designed to defend against known attacks. Different from previous works, we propose an effective adversarial training framework for MGT detectors that builds a generalizable defense against a variety of attacks in the black-box setting.

3 Threat Model

We follow the standard threat modeling framework outlined in prior work (Biggio and Roli, 2018) and describe our assumptions about the adversary's goal and adversary's capability.

Adversary's Goal. Given a piece of MGT, the goal of the adversary is to make trivial changes to the original MGT so as to mislead the prediction of the detector. We refer the changed texts as *adversarial examples*. Ideally, the adversarial examples should satisfy three requirements: *i) Low Perturbation Rate.* Only trivial changes should be applied on the adversarial examples and the semantics of original texts should be retained. *ii) High Readability.* Ad-

versarial examples should exhibit high readability so that the attack is most invisible to humans. *iii) Less Query Requirements.* The adversary should be query-efficient to reduce the time and budget needed to construct each adversarial example.

Adversary's Capability. We assume the adversary's capability in a real-world setting. First, an adversary only maliciously edits the MGTs but would not make any changes to the HWTs. This is because HWTs are trustworthy and there is no need for the adversary to change the prediction on HWTs. Second, since most commercial MGT detectors (*e.g.*, GPT Zero[2]) are close-sourced, the adversary should not have access to model weights and internal states of the target model. The only information the adversary is permitted to query is the output of the detector. Third, the adversary is allowed to access any open-sourced models.

4 Methodology

We introduce the framework of GREATER in this section. The architecture of GREATER is shown in Figure 2. In the following subsections, we first describe the workflow of the adversary and detector, respectively. Then we systematically outline the adversarial training process.

4.1 GREATER-A for Generating Adversarial Examples

To achieve the Adversary's Goal outlined in §3, we developed an effective and efficient adversary. Specifically, the adversary achieves these requirements through two stages: *Identify & Perturb* and *Replace & Refine*.

4.1.1 Identify & Perturb

In this module, we design a token importance estimation module and apply a targeted perturbation on the embeddings of important tokens.

Important Token Identification. We consider a black-box setting where the internal state of the target detector $\mathcal{M}_{tar}(.)$ is inaccessible. Given an original MGT $X = [x_1, x_2, \ldots, x_T]$ consisting of T tokens, we utilize a surrogate model $\mathcal{M}_{sur}(.)$ instead to obtain the last layer hidden state of each token in the text:

$$H = \mathcal{M}_{sur}(X) = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T], \quad (1)$$

where $\mathbf{h}_t$ represents the last layer hidden state of the t-th token x_t generated by $\mathcal{M}_{sur}(.)$. To obtain
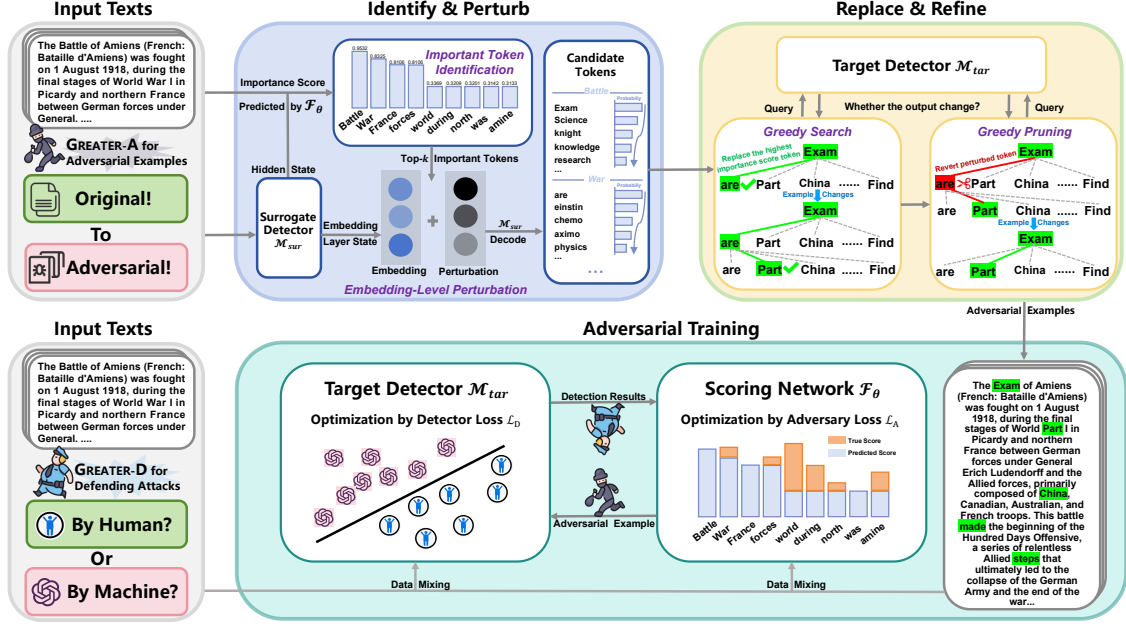
---

[2] https://gptzero.me/

Figure 2: **Pipeline of GREATER.** The adversary identifies important tokens in the original MGT and generates candidates for important tokens (§4.1.1). The adversary conducts and refines the attack by greedy search and pruning (§4.1.2). The final adversarial examples are fed to the target detector (§4.2) and participate in the adversarial training process (§4.3).

the importance score of each token s_t, we train a simple scoring network $\mathcal{F}_\theta(.)$ which takes the feature embeddings as input and outputs the prediction of importance scores for each token:

$$s_t = \mathcal{F}_\theta(\mathbf{h}_t), \qquad (2)$$

where θ are learnable parameters. Then, we select the top-k tokens with the highest importance scores in text X and construct the important-token set $\mathbf{I}$:

$$\mathbf{I} = \text{top-}k\left([(x_t, s_t) \mid t = 1, 2, \ldots, T]\right). \quad (3)$$

To mitigate the impact of discrepancies between the $\mathcal{M}_{sur}(.)$ and $\mathcal{M}_{tar}(.)$, we leverage the predictions of the target detector $\mathcal{M}_{tar}(.)$ to guide $\mathcal{F}_\theta(.)$ in more accurately identifying important tokens during adversarial training process. We detail the adversarial training process in §4.3.

Embedding-level Perturbation. We apply a targeted perturbation on the embedding of the tokens in $\mathbf{I}$ to improve the attack effectiveness while preserving semantic integrity. Formally, we introduce perturbations to the tokens in set $\mathbf{I}$ within the embeddings $E = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_T]$ of the surrogate model to obtain the perturbed embedding $\tilde{E} = [\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, \ldots, \tilde{\mathbf{e}}_T]$:

$$\tilde{\mathbf{e}}_t = \mathbf{e}_t + \mathbf{1}_{[t \in \mathbf{I}]}\delta_t, \qquad (4)$$

where δ_t represents the perturbation of the t-th token, and $\mathbf{1}_{[t \in \mathbf{I}]}$ represents an indicator function with a value of 1 if and only if the condition $t \in \mathbf{I}$ is satisfied, otherwise, it is 0. For the calculation of δ_t, we first initialize the perturbation from a normalized uniform distribution, and then design a single-step gradient ascent strategy to optimize the perturbation. Specifically, we update the perturbation towards the direction where the KL divergence between the output distributions with respect to the original embedding E and initial perturbed embedding $\tilde{E}^0$ increases most steeply. This process is formulated as:

$$\delta_t^0 \sim \mathcal{U}(a,b), \quad \hat{\delta}_t^0 = \xi \frac{\delta_t^0}{\|\delta_t^0\|_2},$$

$$\delta_t = \epsilon \frac{\nabla_{\hat{\delta}_t^0} \text{KL}\left(P_{\text{sur}}(\mathbf{y} \mid E) \parallel P_{\text{sur}}(\mathbf{y} \mid \tilde{E}^0)\right)}{\left\|\nabla_{\hat{\delta}_t^0} \text{KL}\left(P_{\text{sur}}(\mathbf{y} \mid E) \parallel P_{\text{sur}}(\mathbf{y} \mid \tilde{E}^0)\right)\right\|_2}, \qquad (5)$$

where $\hat{\delta}_t^0$ represents the normalized value of δ_t^0, and ξ is a scaling factor. The parameters a and b define the lower and upper bounds of the uniform distribution $\mathcal{U}(a,b)$, ϵ is a scaling factor, $P_{\text{sur}}(\mathbf{y}|E)$ and $P_{\text{sur}}(\mathbf{y}|\tilde{E}^0)$ are label distribution before and after perturbation, respectively.

Afterwards, we project the $\tilde{E}$ back to the vocabulary with the language modeling head and select the top-m tokens with the highest probabilities as

Algorithm 1 Greedy Search Procedure

1: **Input:** Target Detector $\mathcal{M}_{tar}$, Original MGT X, Label c, Important-token Set $\mathbf{I}$.
2: $round \leftarrow 0$ and $\tilde{X} \leftarrow X$.
3: **while** $round < round_{max}$ **do**
4:     Get the token $x_t = \mathbf{I}[round]$ to be perturbed.
5:     Compute corresponding perturbation δ_t using Eq.(5).
6:     Get candidates $\mathbf{C}_t$ for replacing token using Eq.(6).
7:     Replace x_t with the token in $\mathbf{C}_t$ to update $\tilde{X}$.
8:     Classify $\tilde{X}$ via $\mathcal{M}_{tar}$ and obtain the output $\tilde{c}$.
9:     **if** $\tilde{c} \neq c$ **then**
10:         **break**   // Attack success
11:     **else**
12:         $round \leftarrow round + 1$ // Attack failed
13:     **end if**
14: **end while**
15: **Output:** Adversarial Example $\tilde{X}$.

Algorithm 2 Greedy Pruning Procedure

1: **Input:** Adversarial Example $\tilde{X}$ from Algorithm 1, Label c, Important-token Set $\mathbf{I}$ and Target Detector $\mathcal{M}_{tar}$.
2: **for** each perturbed token $\tilde{x}_t$ in $\tilde{X}$ **do**
3:     Revert $\tilde{x}_t$ to corresponding token x_t in X.
4:     Classify $\tilde{X}$ via $\mathcal{M}_{tar}$ and obtain the output $\tilde{c}$.
5:     **if** $\tilde{c} \neq c$. **then**
6:         **continue**   // revert successful
7:     **else**
8:         Replace x_t with $\tilde{x}_t$ again.   // revert failed
9:     **end if**
10: **end for**
11: **Output:** Final Adversarial Example $\tilde{X}$.

---

candidates for replacing the important tokens:

$$\mathbf{C}_t = \text{top-}m(\text{Softmax}(\text{LMHead}(\mathcal{M}_{sur}(\tilde{E}))_t)), \quad (6)$$

where $\text{LMHead}(\mathcal{M}_{sur}(\tilde{E}))_t$ represents the output of the $\text{LMHead}(.)$ at position t. Following POS Constraints (Zhou et al., 2024), which require that the candidate words must match the part of speech of the words they replace, we filter the candidate tokens so that the candidate set contains the tokens with the same POS as the original one.

4.1.2 Replace & Refine

Based on the token importance calculated in §4.1.1, we introduce a greedy search and a greedy pruning strategy to efficiently construct powerful adversarial examples facilitating adversarial training.

Greedy Search. We present the process of greedy search in Algorithm 1. For a piece of original MGT X, we substitute the token x_t with the most possible candidate token in $\mathbf{C}_t$ sequentially according to the descending order of importance scores. After each replacement, we query the target model $\mathcal{M}_{tar}$ if the adversarial example in the current step is machine-generated. We iterate this token-replacing procedure until the adversarial example deceives

the $\mathcal{M}_{tar}$ or all the tokens in $\mathbf{I}$ are replaced. We then use the successful adversarial example (if the attack succeeds) or the text that is perturbed the most as training data for $\mathcal{M}_{tar}$. Compared to existing methods (Liu et al., 2024a; Yu et al., 2024; Hu et al., 2024), our method perturbs only the tokens in set $\mathbf{I}$ incrementally with the guidance of importance score, enabling the generation of adversarial examples with fewer queries and lower perturbation rates.

Greedy Pruning. Due to the local optimality characteristic of greedy search (Yu et al., 2024; Prim, 1957), the adversarial examples constructed by greedy search contain redundant perturbations. We apply the greedy pruning algorithm, as shown in Algorithm 2, to further reduce the perturbation rate and make the attack stealthy without sacrificing its effectiveness. Given an adversarial example $\tilde{X}$ generated with greedy search, we sample the perturbed token by order of importance scores and replace the selected token with its corresponding original token one-by-one. After each restoration, we query the target model $\mathcal{M}_{tar}$ if $\tilde{X}$ is still a successful adversarial example. If the attack remains successful after restoration, the token is converted to the original one; otherwise, we preserve the perturbed token. We loop over all perturbed tokens and produce the final adversarial example $\tilde{X}$.

To further validate the efficiency of our method, we provide a detailed theoretical analysis of query complexity and perturbation rate in GREATER-A in Appendix E.

4.2 GREATER-D for Defending Attacks

Our GREATER-D contains a target detector $\mathcal{M}_{tar}(.)$. For the target detector $\mathcal{M}_{tar}(.)$, we expect it to learn to defend against adversarial attacks from the adversary and generalize the defense ability to other attacks. Specifically, given the target detector $\mathcal{M}_{tar}(.)$, for each original MGT $X_i \in \mathbf{X}$ and the corresponding adversarial example $\tilde{X}_i$ generated by the adversary, which share the same label c_i, the optimization objective of the target detector is to make correct predictions for both the original MGT and the adversarial example:

$$\min_{\theta}(\sum_{X_i \in \mathbf{X}} (\mathcal{L}(\mathcal{M}_{tar}^{\theta}(X_i), c_i) + \mathcal{L}(\mathcal{M}_{tar}^{\theta}(\tilde{X}_i), c_i))), \quad (7)$$

where θ represents the learnable parameters of the target detector $\mathcal{M}_{tar}(.)$ and $\mathcal{L}$ is the loss function. This optimization objective aims to guide

the $\mathcal{M}_{tar}(.)$ to simultaneously enhance its performance on both original MGT and adversarial examples, thereby compelling it to learn robust features of samples before and after attacks. Regardless of whether the samples are subjected to adversarial interference, these features ensure that the detector can accurately classify the samples. As a result, the detector is better equipped to handle various types of attacks.

4.3 Adversarial Training

We propose to train the adversary and detector in a co-training manner, that is, the two main components in GREATER are updated in the same training step. Unlike the previous methods (Zhang et al., 2024b; Huang et al., 2024; Zeng et al., 2023) that rely on static training set augmented with adversarial examples, synchronously updating the adversary and the detector allows the detector to learn from easy adversarial examples to hard ones, facilitating the defense to generalize to different attacks.

Adversary Loss. The goal of the adversary is to precisely estimate the importance of each token in the detection to guide it to undertake a successful attack. Thus, we use the gradient with respect to the input in calculating the cross-entropy loss on training data as the golden label for the token importance score. However, since we are constrained in a black-box setting, we utilize the $\mathcal{M}_{sur}(.)$ to obtain the gradient. This process is formulated as:

$$s_x^* = \left\| \nabla_x \mathcal{L}_{\mathrm{sur}} \right\|_2,$$

$$\mathcal{L}_{\mathrm{sur}} = \frac{1}{M} \sum_{i=1}^{M} \left[-\log P_{\mathrm{sur}}(c_i \mid X_i) \right], \quad (8)$$

where $\mathcal{L}_{\mathrm{sur}}$ is the cross-entropy classification loss of the surrogate model to the original MGTs. Subsequently, we update the scoring network $\mathcal{F}_\theta(.)$ with the mean squared error loss:

$$\mathcal{L}_{\mathrm{imp}} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|X_i|} \sum_{x \in X_i} (s_x - s_x^*)^2, \quad (9)$$

where $|X_i|$ is the number of tokens in sample X_i. In addition, we leverage the output information from the target detector to guide the training of the $\mathcal{F}_\theta(.)$, thereby mitigating the impact of discrepancies between the surrogate and target detector. Specifically, we replace the true labels in the cross-entropy loss with misleading labels to direct the

$\mathcal{F}_\theta(.)$ to produce samples that are capable to deceive the detector:

$$\mathcal{L}_{\mathrm{adv}} = \frac{1}{M} \sum_{i=1}^{M} \left[-\log P_{\mathrm{tar}}(1 - c_i | \tilde{X}_i) \right]. \quad (10)$$

Finally, we balance the influence of various losses on the adversary by adjusting the weight parameter λ. The total loss for the adversary is given as follows:

$$\mathcal{L}_{\mathrm{A}} = \lambda \, \mathcal{L}_{\mathrm{adv}} + (1 - \lambda) \, \mathcal{L}_{\mathrm{imp}}. \quad (11)$$

Detector Loss. The detector's goal is to maintain correct predictions in all circumstances. Based on this, we optimize the detector towards minimizing a cross-entropy loss:

$$\mathcal{L}_{\mathrm{D}} = \frac{1}{M} \sum_{i=1}^{M} \left[-\log P_{\mathrm{tar}}(c_i | X_i) - \log P_{\mathrm{tar}}(c_i | \tilde{X}_i) \right], \quad (12)$$

where $P_{\mathrm{tar}}(c_i | X_i)$ is the probability of the target detector output at the correct label c_i.

Training Process. We update the adversary and the detector alternatively in the same training step. Specifically, at each training step t, we first update the adversary with loss function (10) while keeping the detector frozen. The updated adversary generates adversarial example $\tilde{X}_i$ in the current step. Then we update the detector with loss function (12). We detail the adversarial training process in form of pseudocode in Appendix F.

5 Experiment Results

We conduct extensive experiments to comprehensively evaluate the defense performance of our GREATER-D and also reveal the vulnerability of the current defense strategy with the adversarial examples generated by GREATER-A.

5.1 Defense Performance for GREATER-D

Experiment Setting. We evaluate our defense model GREATER-D against 16 text perturbation and adversarial attack methods, whose detailed introductions are outlined in the Appendix A.5. The competitors include **i) data augmentation methods:** Editing Pretrained (EP) (Wang et al., 2024c), Paraphrasing Pretrained (PP) (Wang et al., 2024c), CERT-ED (Huang et al., 2024), RanMask (Zeng et al., 2023), Text-RS (Zhang et al., 2024b), Text-CRS (Zhang et al., 2024a). **ii) adversarial training methods:** Virtual Adversarial Training (VAT) (Miyato et al., 2016), Token Aware Virtual Adversarial Training (TAVAT) (Li et al., 2021), RADAR

3096

Category	Method	Metric	Baseline	Adversarial Data Augmentation						Adversarial Training				
			F.t.XLM-RoBERTa-Base	EP	PP	CERT-ED	RanMask	Text-RS	Text-CRS	VAT	TAVAT	RADAR	OUTFOX	GREATER-D
Text Perturbation	*Mixed Edit*	ASR(%)↓	34.65	4.58	32.33	16.74	17.00	22.81	15.34	33.19	37.50	18.76	11.33	8.85
	HMGC	ASR(%)↓	30.00	6.53	27.34	12.52	11.74	16.58	13.37	25.19	31.50	8.53	3.74	2.28
	Paraphrasing	ASR(%)↓	70.58	54.65	6.27	32.10	40.20	59.58	26.67	67.98	65.90	2.08	14.96	3.45
	*Code-switching MF**	ASR(%)↓	50.58	38.37	34.08	29.19	27.78	48.80	27.15	36.71	47.52	32.57	5.46	1.13
	*Code-switching MR**	ASR(%)↓	47.91	31.23	30.21	5.30	10.80	16.98	14.36	38.03	46.46	14.14	5.37	1.02
	Human Obfuscation	ASR(%)↓	18.42	22.19	27.00	13.64	18.86	18.05	15.24	25.35	24.17	16.26	5.74	0.86
	Emoji-cogen	ASR(%)↓	32.19	46.55	44.17	11.41	22.23	17.72	27.10	52.35	40.33	33.13	7.72	0.47
	Typo-cogen	ASR(%)↓	60.10	61.57	59.72	27.29	38.82	37.09	44.79	70.26	63.04	41.52	9.29	1.08
	ICL	ASR(%)↓	1.40	1.41	1.30	1.72	0.83	1.89	0.67	1.13	1.88	2.77	0.59	0.20
	Prompt Paraphrasing	ASR(%)↓	0.00	0.69	0.00	0.70	0.00	0.00	0.00	0.00	0.00	0.94	0.65	0.23
	CSGen	ASR(%)↓	25.44	22.81	6.14	10.65	1.70	2.41	23.95	26.88	27.52	17.07	3.40	0.00
	Avg.	ASR(%)↓	33.75	26.42	24.41	14.66	17.27	21.99	18.97	34.28	35.07	17.07	6.20	1.78
Adversarial Attack	*PWWS*	ASR(%)↓	61.45	48.47	49.06	11.07	14.83	16.13	19.15	53.56	62.68	12.58	-	5.91
		Queries↑	1197.95	1237.62	1235.01	1371.28	1361.15	1356.67	1342.15	1226.53	1180.87	1366.85	-	1390.69
	TextFooler	ASR(%)↓	72.29	70.76	70.02	11.07	17.43	19.56	22.98	69.25	94.02	14.46	-	6.11
		Queries↑	690.92	724.23	713.77	1362.17	1291.35	1271.34	1237.18	780.51	440.28	1301.84	-	1396.52
	BERTAttack	ASR(%)↓	71.49	69.34	63.52	6.04	12.42	12.30	16.94	69.45	93.81	13.08	-	5.70
		Queries↑	411.54	446.64	425.68	710.56	682.29	680.14	667.82	450.57	279.02	677.24	-	718.84
	A2T	ASR(%)↓	45.47	37.45	50.10	6.24	11.22	14.52	14.31	29.12	54.64	8.58	-	5.09
		Queries↑	293.26	320.88	302.18	516.07	499.89	493.25	488.91	361.88	207.15	505.54	-	519.77
	T-PGD	ASR(%)↓	49.90	34.15	25.28	8.52	12.07	13.15	14.92	42.14	45.58	7.79	-	5.61
		Queries↑	354.75	515.62	712.08	799.62	766.53	752.04	744.17	404.85	377.69	813.76	-	824.58
	GREATER-A(ours)	ASR(%)↓	96.58	87.08	84.34	62.17	63.58	75.25	82.29	89.26	85.02	71.04	-	46.08
		Queries↑	62.63	66.71	68.53	99.56	98.92	106.08	75.98	63.57	67.17	100.74	-	190.64
	Avg.	ASR(%)↓	66.20	57.87	57.05	17.52	21.92	25.15	28.43	58.80	72.62	21.26	-	12.42
		Queries↑	501.84	551.95	576.21	809.88	783.36	776.59	759.37	547.98	425.36	794.33	-	840.17
Total	*Avg.*	ASR(%)↓	45.20	37.52	35.93	15.67	18.91	23.11	22.31	42.93	48.33	18.55	6.20	5.53

Table 1: **Performance of defense methods under different attacks.** The best results are highlighted in green background. ∗ means that Code-switching MF and Code-switching MR are two variations of Code-switching method. Since the detector in OUTFOX is a closed-source LLM, it cannot be attacked by adversarial attack. Therefore, we only evaluate OUTFOX under text perturbation attacks.

(Hu et al., 2023a), OUTFOX (Koike et al., 2024). Detailed introduction and implementation are presented in Appendix A.4.1 and A.2. The dataset we use is presented in Appendix A.1.

Experiment results. We present the defense performance of GREATER-D in Table 1 and unveil the following three key insights: 1) **Best defense performance.** Our method exhibits the best defense performance against different attacks among all competitors. The average Attack Success Rate (ASR) drops to **1.78%** for text perturbation attacks and **12.42%** for adversarial attacks, which are lower than the second-best methods, **RADAR (6.20%)** and **CERT-ED (17.52%)**, by **4.42%** and **5.10%**, respectively. 2) **Most effort needed for adversarial attack.** We observe that it takes adversarial attack more resources to conduct a successful attack to GREATER-D. **840.17** queries are required on average, which is **30.29** more than other defense methods. Moreover, the average ASR against GREATER-D is only **12.42%** for adversarial attacks. It illustrates that GREATER-D makes adversarial attacks both inefficient and in-

effective. 3) **Generalized Defense to Different Attacks.** We notice that GREATER-D significantly reduces the ASR of different kinds of attacks even though it is trained with GREATER-A. As an exemplary method, EP performs better than GREATER-D when defending Mixed Edit Attack but cannot withstand other attacks. The defense against a wide variety of attacks demonstrates the generalized defense effect of GREATER-D.

5.2 Attack Performance for GREATER-A

In this section, we evaluate the effectiveness of GREATER-A in the black-box setting using the metrics detailed in Appendix A.3. We categorize the comparison methods into two classes: i) *Query-based methods*, which query the target model for output to adjust attack strategy, including PWWS (Ren et al., 2019), TextFooler (Jin et al., 2020), BERTAttack (Li et al., 2020), HQA (Liu et al., 2024a), ABP (Yu et al., 2024), T-PGD (Yuan et al., 2023), and FastTextDodger (Hu et al., 2024). ii) *Zero-query methods*, which conducts attack without any information from the target model, in-

cluding WordNet (Zhou et al., 2024), Back Translation (Zhou et al., 2024), Rewrite (Zhou et al., 2024), T-PGD (Yuan et al., 2023), and HMGC (Zhou et al., 2024). To further demonstrate the effectiveness of GREATER-A, we also incorporate A2T (Yoo and Qi, 2021), a SOTA white-box method in query-based methods. Detailed introduction of these attack methods are listed in Appendix A.4.2. Note that GREATER-A is a query-based method. However, for a fair comparison, we also implement our method in a zero-query setting where we query the surrogate model for feedback. Among all the attacks, we employ a fine-tuned XLM-RoBERTa-Base model (Conneau et al., 2019) as the target detector.

Experiment Results. We show the experiment results in Table 2. We find GREATER-A performs the best in three dimensions: effectiveness, efficiency, and stealthy. GREATER-A achieves 96.58% and 69.11% in terms of ASR in query and zero-query settings, respectively, which significantly outperforms all other methods. Moreover, in the query-based setting, it only takes 62.63 queries for GREATER-A to conduct a successful attack, which is four times fewer than its competitors. As for the stealthy, the texts edited by GREATER-A achieves the lowest perplexity and has the best readability implied by the highest USE and lowest readability change in query-based scenario. In the zero-query setting, GREATER-A performs second-best in terms of readability after Back Translation which can rarely conduct a successful attack.

Attack Type	Method	Avg Queries ↓	ASR (%) ↑	Pert. (%) ↓	ΔPPL ↓	USE ↑	Δr ↓
	PWWS	1197.95	61.45	4.71	37.85	0.9488	12.76
	TextFooler	690.92	72.29	6.26	46.89	0.9302	21.07
	BERTAttack	411.54	71.49	5.79	36.15	0.9402	15.78
	HQA	283.89	88.13	23.57	102.87	0.8854	72.16
Query-based	FastTextDodger	745.75	63.78	13.29	76.15	0.9188	55.14
	ABP	785.18	75.65	14.63	39.61	0.8709	26.40
	T-PGD	354.75	49.90	38.01	181.31	0.8197	35.09
	GREATER-A	62.63	96.58	7.26	35.22	0.9506	9.21
	A2T (White Box)	293.26	45.47	7.01	62.84	0.9215	32.07
	WordNet	-	42.60	-	26.27	0.90	4.26
	Back Translation	-	2.40	-	6.40	0.91	3.06
Zero-query	Rewrite	-	36.47	-	92.08	0.79	15.62
	T-PGD	-	62.40	-	140.13	0.82	44.99
	HMGC	-	30.00	-	12.94	0.84	36.90
	GREATER-A	-	69.11	-	43.11	0.92	4.55

Table 2: **Attack results of the query and zero-query attack methods on the target model.** Note that perturbation rates are not reported for zero-query methods because the zero-query methods rewrite the whole text. The best result in each group is highlighted with a green background.
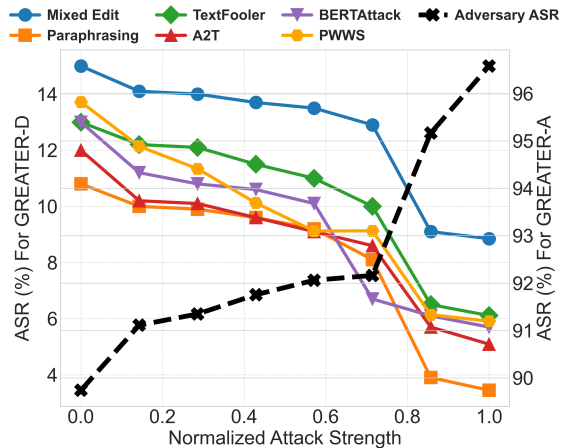

Figure 3: **Impact of attack strength in GREATER.** We normalize the attack strength for better visualization of the results.

6 Discussion

6.1 Impact of Attack Strength of GREATER-A on GREATER-D

In this section, we investigate how the strength of adversarial training affects the performance of both our GREATER-D and GREATER-A. We define the attack strength as the max number of query the adversary is allowed to make in the training process. Generally, if the adversary queries the target model more frequently, its final output tends to be more effective. We increase attack strength in GREATER and evaluate the performance of GREATER-D under 7 attacks and GREATER-A on the fine-tuned XLM-RoBERTa-Base detector and present the results in Figure 3. The ASR significantly increases as the attack strength grows, which proves the rationale of the attack strength measure. We observe that the GREATER-D becomes more robust with the increasing of attack strength, indicated by the decreasing ASR under all attacks. Notably, the ASR under *Paraphrasing* Attack decreases from **10.80** to **3.45** as the attack strength increases from **0.0** to **1.0**, which is **3.13** times lower.

The experimental results indicate that attack strength is a key factor influencing the robustness of the MGT detector. However, an increased number of queries comes with more cost on time and budget. Thus, there exists a trade-off between the effectiveness and efficiency in adversarial training.

6.2 Defense Adaptation to Different Backbones

To demonstrate the generalizability and effectiveness of our GREATER across different model ar-

chitectures, we replace the backbone model with seven state-of-the-art transformer-based models, including both base and large variants of AL-BERT (Lan et al., 2019), DeBERTa (He et al., 2020), RoBERTa (Liu et al., 2019), and XLM-RoBERTa (Conneau et al., 2019). We report the ASR of each model under the Paraphrasing Attack with a budget of 0.74, both before defense (Baseline) and after applying our GREATER-D. We compare it with the best-performing defense method CERT-ED and show the result in Table 3.

Model	Metric	Baseline	CERT-ED	GREATER-D
ALBERT Base (12M)	ASR(%)↓	63.58	46.71	35.62
ALBERT Large (18M)	ASR(%)↓	97.14	45.45	43.40
DeBERTa Base (86M)	ASR(%)↓	22.95	29.37	4.10
DeBERTa Large (304M)	ASR(%)↓	51.72	25.07	10.25
RoBERTa Base (125M)	ASR(%)↓	81.88	26.78	15.32
RoBERTa Large (355M)	ASR(%)↓	30.21	34.66	5.38
XLM-RoBERTa Large (561M)	ASR(%)↓	81.99	40.04	1.73

Table 3: **Experiment results of various models before and after applying GREATER-D under Paraphrasing attack.** The best result in each group is highlighted with a green background.

The results presented in Table 3 demonstrate the efficacy of our adversarial training method across a diverse set of transformer-based models. Notably, *XLM-RoBERTa-Large* exhibits the most substantial improvement, with ASR decreasing by 80.26%, 38.31% compared with baseline and CERT-ED, highlighting the significant impact of adversarial training on models with initially lower performance metrics. Similarly, both *RoBERTa Base* and *DeBERTa Large* demonstrate substantial improvements. Specifically, *RoBERTa Base* achieves a 66.56% reduction in ASR compared to the baseline and outperforms the CERT-ED with an additional 11.46% decrease. Likewise, *DeBERTa Large* exhibits a 41.47% drop in ASR relative to the baseline and surpasses CERT-ED by reducing ASR by an additional 14.82%. These results underscore the robustness and versatility of our adversarial training approach across different model sizes and architectures. While smaller models like *ALBERT Base* and *ALBERT Large* exhibit more modest gains, the consistent upward trends across all evaluated models affirm that our adversarial training method effectively enhances model resilience and performance against Paraphrasing Attack. This versatility makes our approach a valuable tool for improving a wide range of transformer-based models in adversarial settings.

6.3 Impact of Surrogate Model of GREATER-A

We use different surrogate model and dataset for training GREATER-A to demonstrate that GREATER-A's performance is independent of surrogate model selection and training data. Specifically, we use RoBERTa-Large and GPT2 as surrogate models, and use SemEval (same dataset for training GREATER-D, abbreviated as IND) and M4 (Wang et al., 2024b) (different dataset from training GREATER-D, abbreviated as IND) to validate our claim. The results are shown in Table 4.

Surrogate Model	Dataset (Type)	Avg Queries ↓	ASR (%) ↑	Pert. (%) ↓	ΔPPL ↓	USE ↑	Δr ↓
RoBERTa-Large	SemEval (IND)	62.63	96.58	7.26	35.22	0.9506	9.21
RoBERTa-Large	M4 (OOD)	65.77	95.72	7.33	38.84	0.9503	9.65
GPT2	SemEval (IND)	65.60	96.17	7.29	36.87	0.9550	12.73
GPT2	M4 (OOD)	63.80	96.36	7.27	37.14	0.9536	10.64

Table 4: **The attack results of different surrogate models and datasets that the surrogate models are trained with.** The best result in each group is highlighted with a green background.

As shown in Table 4, the attack effectiveness remains stable regardless of the surrogate model or training dataset. This consistency in attack success rates indicates that GREATER-A's performance is not sensitive to changes in the surrogate model or training data, supporting the claim that the use of a surrogate model from the same family as the target model does not leak crucial information.

7 Conclusion

In this paper, we proposed an adversarial training framework **GRE**edy **A**dversary Promo**T**ed Defend**ER** (GREATER) to enhance the robustness of MGT detector under different text perturbation and adversarial attacks. We design a novel attack strategy for the adversary including Identify & Perturb and Replace & Refine to construct effective adversarial examples efficiently. In GREATER, we update the adversary and the detector alternatively in the same training step for better defense generalization. Our experiment results demonstrate the efficacy of our detector GREATER-D under 16 attacks along with the leading performance of adversary GREATER-A compared with 13 methods. The discussion on the relationship between attack strength and defense performance reveals the importance of a powerful adversary in adversarial training for robust MGT detectors.

Limitations

Despite the promising results achieved by GREATER, there are three primary limitations to this study. **First**, our defense method can generalize to different attacks but its application on texts in different languages remains a challenge. **Second**, the computational cost of training the adversarial framework, particularly for the adversary and detector, is substantial, requiring significant hardware resources that could limit its deployment in resource-constrained settings. **Third**, the detector presented in our work is only able to attribute the origin of the text on the document-level. However, more works are needed for fine-grained (*e.g.*, sentence-level, token-level) detection in Human-AI co-authored texts.

Ethics Statement

This study seeks to improve the robustness and security of machine-generated text (MGT) detection, with a focus on defending against adversarial threats. While the proposed GREATER framework enhances detection capabilities, it also introduces potential risks of misuse, such as creating adversarial examples to evade detection systems. To mitigate such risks, our experiments were conducted in controlled environments, and details that could enable misuse were abstracted. We emphasize that this work is intended solely for advancing detection technologies and defending against malicious applications. Ethical use of these findings is imperative, and any misuse for harmful purposes is strongly discouraged. The artifacts used in our work are all under the restriction of the license.

Acknowledgment

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude 3: A conversational ai model.

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

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2023. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. *arXiv preprint arXiv:2310.05130*.

Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding nlp systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647.

Vitalii Fishchuk. 2023. Adversarial attacks on neural text detectors. B.S. thesis, University of Twente.

Rudolf Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

Evgeniy Gabrilovich and Alex Gontmakher. 2002. The homograph attack. *Communications of the ACM*, 45(2):128.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xun Guo, Yongxin He, Shan Zhang, Ting Zhang, Wanquan Feng, Haibin Huang, and Chongyang Ma. Detective: Detecting ai-generated text via multi-level contrastive learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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. *arXiv preprint arXiv:2401.12070*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Helsinki-NLP. 2020. Helsinki-nlp machine translation models. https://huggingface.co/Helsinki-NLP.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023a. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095.

Xiaoxue Hu, Geling Liu, Baolin Zheng, Lingchen Zhao, Qian Wang, Yufei Zhang, and Minxin Du. 2024. Fast-textdodger: Decision-based adversarial attack against black-box nlp models with extremely high efficiency. *IEEE Transactions on Information Forensics and Security*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023b. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*.

Zhuoqun Huang, Neil G Marchant, Olga Ohrimenko, and Benjamin IP Rubinstein. 2024. Cert-ed: Certifiably robust text classification for edit distance. *arXiv preprint arXiv:2408.00728*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2024. Outfox: Llm-generated essay detection through in-context learning with adversarially generated examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21258–21266.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM computing surveys (CSUR)*, 24(4):377–439.

Laida Kushnareva, Tatiana Gaintseva, German Magai, Serguei Barannikov, Dmitry Abulkhanov, Kristian Kuznetsov, Eduard Tulchinskii, Irina Piontkovskaya, and Sergey Nikolenko. 2024. Ai-generated text boundary detection with roft. In *1st Conference on Language Modeling (COLM)*, volume 2024.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Linyang Li, Ruotian Ma, Xiaonan Guo, Qing Wang, Xipeng Qiu, and Xuanjing Tang. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Yuan Li, Shuhuai Ren, and Zhouxing Shi. 2021. Tavat: Token-aware virtual adversarial training for language understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2924–2936.

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023. Gpt detectors are biased against non-native english writers. *Patterns*, 4(7).

Han Liu, Zhi Xu, Xiaotong Zhang, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. 2024a. Hqa-attack: toward high quality black-box hard-label adversarial attack on text. *Advances in Neural Information Processing Systems*, 36.

Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. 2024b. Does detectgpt fully utilize perturbation? bridging selective perturbation to fine-tuned contrastive learning detector would be better. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1889.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2022. Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. *arXiv preprint arXiv:2212.10341*.

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

Dominik Macko, Robert Moro, Adaku Uchendu, Ivan Srba, Jason Samuel Lucas, Michiharu Yamashita, Nafis Irtiza Tripto, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2024. Authorship obfuscation in multilingual machine-generated text detection. *arXiv preprint arXiv:2401.07867*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2016. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993.

Robert Clay Prim. 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 1(8):1–24.

Shuhuai Ren, Yihe Zheng, Yizhan Chen, Bin Yu, Zhiyuan Liu, and Maosong Sun. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. 2024. Red teaming language model detectors with language models. *Transactions of the Association for Computational Linguistics*, 12:174–189.

M. Siino. 2024. Badrock at semeval-2024 task 8: Distilbert to detect multigenerator, multidomain and multilingual black-box machine-generated text. In *Proceedings of the 18th International Workshop on Semantic Evaluation*.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 8384–8395. Association for Computational Linguistics (ACL).

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2024. Ghostbuster: Detecting text ghostwritten by large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1702–1717.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023. Seqxgpt: Sentence-level ai-generated text detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1144–1156.

Yichen Wang, Shangbin Feng, Abe Bohan Hou, Xiao Pu, Chao Shen, Xiaoming Liu, Yulia Tsvetkov, and Tianxing He. 2024a. Stumbling blocks: Stress testing the robustness of machine-generated text detectors under attacks. *arXiv preprint arXiv:2402.11638*.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, et al. 2024b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407.

Zaitian Wang, Pengfei Wang, Kunpeng Liu, Pengyang Wang, Yanjie Fu, Chang-Tien Lu, Charu C. Aggarwal, Jian Pei, and Yuanchun Zhou. 2024c. A comprehensive survey on data augmentation. *arXiv preprint arXiv:2405.09591*.

Genta Winata, Alham Fikri Aji, Zheng Xin Yong, and Thamar Solorio. 2023. The decades progress on code-switching research in nlp: A systematic survey on trends and challenges. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2936–2978.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of nlp models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1001–1013.

Zhen Yu, Zhenhua Chen, and Kun He. 2024. Query-efficient textual adversarial example generation for black-box attacks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 556–569.

Lifan Yuan, Yichi Zhang, Yangyi Chen, and Wei Wei. 2023. Bridge the gap between cv and nlp! a gradient-based textual adversarial attack framework. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7132–7146.

JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.

Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2023. Certified robustness to text adversarial attacks by randomized [mask]. *Computational Linguistics*, 49(2):345–373.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020a. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International conference on machine learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xinyu Zhang, Hanbin Hong, Yuan Hong, Peng Huang, Binghui Wang, Zhongjie Ba, and Kui Ren. 2024a. Text-crs: A generalized certified robustness framework against textual adversarial attacks. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy*, pages 53–53.

Zeliang Zhang, Wei Yao, Susan Liang, and Chenliang Xu. 2024b. Random smooth-based certified defense against text adversarial attack. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1251–1265.

Wanjun Zhong, Duyu Tang, Zenan Xu, Ruize Wang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Neural deepfake detection with factual structure of text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2461–2470.

Ying Zhou, Ben He, and Le Sun. 2024. Humanizing machine-generated content: Evading ai-text detection through adversarial attack. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8427–8437.

A Experimental Setting

A.1 Dataset

We used the Semeval Task8 dataset (Siino, 2024) as the primary dataset for training the detector. This large-scale dataset includes MGTs from ChatGPT 4, Davinci, Bloomz, Dolly, and Cohere, as well as HWTs from WikiHow, Reddit, arXiv, Wikipedia, and PeerRead. Moreover, the average token length of the dataset is 623.2521. For each scenario, we employed distinct datasets for testing, as detailed in Tabel 5.

Scenario	Dataset	Number of MGTs
Mixed Edit	Semeval Task8 (Siino, 2024)	1000
HMGC	Semeval Task8 (Siino, 2024)	1000
Paraphrasing	Semeval Task8 (Siino, 2024)	1000
Code-switching	Semeval Task8 (Siino, 2024)	1000
Human Obfuscation	Semeval Task8 (Siino, 2024)	1000
Emoji-cogen	Wang et al. (Wang et al., 2024a)	500
Typo-cogen	Wang et al. (Wang et al., 2024a)	500
ICL	Wang et al. (Wang et al., 2024a)	500
Prompt Paraphrasing	Wang et al. (Wang et al., 2024a)	500
CSGen	Wang et al. (Wang et al., 2024a)	500
Adversarial Attack	Semeval Task8 (Siino, 2024)	500

Table 5: Experimental scenarios and corresponding datasets.

A.2 Implementation

GREATER is deployed on a server equipped with 4 NVIDIA A100 GPUs, running on Ubuntu 22.04. The adversarial framework uses the xlm-roberta-base model (279M) as its base detector. For the evaluation of Mixed Edit Attack, Paraphrasing Attack, Code-switching Attack, and Human Obfuscation, we adopt the concept of "budget" to control the intensity of the attacks for a more fine-grained investigation of model robustness, following the methodology in Wang et al. (2024a). Specifically, Mixed Edit Attack uses character edit distance (Levenshtein, 1966) as the budget, Paraphrasing Attack and Code-switching Attack utilize BERTSCore (Zhang et al., 2019) as the budget, and Human Obfuscation Attack employs the confusion ratio as the budget.

For all defense methods, we use xlm-roberta-base as the base detector. The sizes of the training set, validation set, and test set are 8000, 1000, and 1000, respectively. The learning rate is set to 1e-5, and the number of epochs is fixed at 6.

For all data augmentation-based methods, we use 20% of MGT for data augmentation. If the method has hyperparameters, they are set according to the reference values in the original paper. For all adversarial training-based methods, we use 20% of MGT for adversarial training. If the method has hyperparameters, they are set according to the reference values in the original paper. For our method, our detector is trained using a label smoothing loss function with a smoothing factor of α. We use a trained RoBERTa Large as the Surrogate Model $\mathcal{M}_{\text{sur}}(.)$ due to its strong generalization ability and precise understanding of English text. Our method's selected hyperparameters are shown in Table 6.

Hyperparameter	Value
Weight Parameter λ	0.05
Scaling Factor ϵ	0.3
Scaling Factor ξ	0.01
Lower Bound of the Uniform Distribution a	0.5
Upper Bound of the Uniform Distribution b	-0.5
Batch Size M	50
Epoch N	6
Label Smoothing Factor α	0.1

Table 6: Hyperparameters for our GREATER.

A.3 Evaluation Metrics

We use attack effectiveness metrics and text quality metrics to comprehensively evaluate the performance of defense and attack methods.

1) Attack Effectiveness Metrics

Attack Success Rates (ASR). The Attack Success Rate (ASR) measures the proportion of successful attacks relative to the total number of attempted attacks. Note that we only attack text that was detected as machine-written before the attack. ASR is calculated as follows:

$$\text{ASR} = \frac{\text{Text detected as HWT after attack}}{\text{Text detected as MGT before attack}}.$$

For detector, a lower ASR indicates better defense performance, while for adversary, a higher ASR signifies a stronger attack effectiveness.

2) Text Quality Metrics

Perturbation Rate (Pert.). Pert. measures the lexical difference between the adversarial text and the original text. It is defined as the ratio of the number of perturbed tokens to the total number of tokens in the text. A lower perturbation rate

indicates that the adversarial text remains more similar to the original text.

Perplexity Variation (ΔPPL)**.** ΔPPL measures the change of perplexity, which represents the consistency and fluency of the adversarial examples. The PPL is calculated as:

$$\text{PPL} = exp(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, \ldots, w_{i-1})),$$

where N is the total number of tokens, w_i represents the i-th token, and $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is the probability assigned by the language model. Generally, a lower PPL variation indicates that the quality of the adversarial text is closer to that of the original text. We use GPT-2 (Radford et al., 2019) to compute PPL.

Universal Sentence Encoder (USE) score(Cer et al., 2018)**.** USE evaluates the semantic similarity between the adversarial example and the original text. The USE score is computed as:

$$\text{USE Score} = \frac{\mathbf{E}_{\text{orig}} \cdot \mathbf{E}_{\text{adv}}}{\|\mathbf{E}_{\text{orig}}\|\|\mathbf{E}_{\text{adv}}\|},$$

where $\mathbf{E}_{\text{orig}}$ and $\mathbf{E}_{\text{adv}}$ are the sentence embeddings of the original and adversarial texts, respectively, generated by the USE model, and $\|\mathbf{E}\|$ denotes the Euclidean norm. A higher USE score indicates greater semantic similarity. We use USE (Universal Sentence Encoder) to calculate USE score.

Flesch Reading Ease score (Δr) (Flesch, 1948)**.** Δr measures the variation in text readability. The Flesch Reading Ease score is calculated as:

$$r = 206.835 - 1.015 \times \frac{N_{\text{words}}}{N_{\text{sentences}}} - 84.6 \times \frac{N_{\text{syllables}}}{N_{\text{words}}},$$

where N_{words} is the total number of words in the text, $N_{\text{sentences}}$ is the total number of sentences, and $N_{\text{syllables}}$ is the total number of syllables. A smaller Δr indicates less change in readability.

A.4 Experimental Comparison Methods

This section provides a detailed introduction to the SOTA methods included in our comparisons.

A.4.1 Defense Methods

Edit Pretraining (EP) (Wang et al., 2024c): An adversarial data augmentation method that blends Mixed Edit Attack into the training set.

Paraphrasing Pretraining (PP) (Wang et al., 2024c): An adversarial data augmentation method that blends Paraphrasing Attack into the training set.

CERT-ED (Huang et al., 2024): An adversarial data augmentation method that adapts randomized deletion to effectively safeguard natural language classification models from diverse edit-based adversarial operations, including synonym substitution, insertion, and deletion.

RanMask (Zeng et al., 2023): An adversarial data augmentation method that randomly masks a proportion of words in the input text, thereby mitigating both word- and character-level adversarial perturbations without assuming prior knowledge of the adversaries' synonym generation.

Text-RS (Zhang et al., 2024b): An adversarial data augmentation method that treats discrete word substitutions as continuous perturbations in the embedding space to reduce the complexity of searching through large discrete vocabularies and bolster the model's robustness.

Text-CRS (Zhang et al., 2024a): An adversarial data augmentation method that is built on randomized smoothing, encompassing various word-level adversarial manipulations—such as synonym substitution, insertion, deletion, and reordering—by modeling them in both embedding and permutation spaces.

Virtual Adversarial Training (VAT) (Miyato et al., 2016): An Adversarial Training method that defines the adversarial direction without label information and employs the robustness of the conditional label distribution around each data point against local perturbation as the adversarial loss.

Token Aware Virtual Adversarial Training (TAVAT) (Li et al., 2021): An Adversarial Training method that uses token-level accumulated perturbation to better initialize the noise and applies token-level normalization.

RADAR (Hu et al., 2023a): An Adversarial Training method that uses a paraphraser (such as DIPPER) as adversary and a fine-tuned model as detector. The adversary and detector learn from each other and improve the robustness of the detector when facing paraphrasing attack.

OUTFOX (Koike et al., 2024): An adversarial training method that iteratively improves a detector by generating in-context adversarial examples via a co-evolving attacker. The attacker crafts examples to evade the detector, which are then used to strengthen the detector through in-context learning.

A.4.2 Attack Methods

TextFooler (Jin et al., 2020): A black-box attack method targeting text classification and natural language inference tasks. It ranks words by their importance to the model's prediction and replaces them with semantically similar and grammatically correct synonyms to generate adversarial examples while preserving sentence meaning and structure.

BERTAttack (Li et al., 2020): A black-box attack method that utilizes a pre-trained BERT model to generate adversarial examples. It replaces certain words in the target sentence with high-probability candidates predicted by BERT, ensuring the adversarial examples remain semantically and grammatically similar to the original while misleading the model.

PWWS (Ren et al., 2019): A black-box attack method designed for text classification models. It computes the saliency of each word in the sentence, ranks them accordingly, and replaces high-saliency words with synonyms from WordNet, generating adversarial examples that mislead the model while preserving meaning.

A2T (Yoo and Qi, 2021): A white-box attack method aimed at improving adversarial training. It leverages gradient-based word importance estimation, performing a greedy search from least to most important words, and uses word embedding models or masked language models to generate candidate replacements, ensuring adversarial examples remain semantically similar while misleading the model.

HQA (Liu et al., 2024a): A black-box adversarial attack method for text classification task. It initializes adversarial examples by minimizing perturbations and iteratively substitutes words using synonym sets to optimize both semantic similarity and adversarial effectiveness while reducing query consumption.

ABP (Yu et al., 2024): A black-box adversarial attack method leveraging prior knowledge to guide word substitutions efficiently. It introduces Adversarial Boosting Preference (ABP) to rank word importance and proposes two query-efficient strategies: a query-free attack (ABPfree) and a guided search attack (ABPguide), significantly reducing query numbers while maintaining high attack success rates.

T-PGD (Yuan et al., 2023): A black-box zero-query adversarial attack method extending optimization-based attack techniques from computer vision to NLP. It applies perturbations to the embedding layer and amplifies them through forward propagation, then uses a masked language model to decode adversarial examples.

FastTextDodger (Hu et al., 2024): A black-box adversarial attack designed for query-efficient adversarial text generation. It generates grammatically correct adversarial texts while maintaining strong attack effectiveness with minimal query consumption.

A.5 Detailed Information of Text Perturbation Method

In this section, we introduce 10 Text Perturbation Methods mentioned in Table 1. Detailed information of Adversarial Attack Methods can be found in Appendix A.4.2.

Mixed Edit (Wang et al., 2024a): A text modification strategy combining homograph substitution, formatting edits, and case conversion to evade detection. It manipulates character representation, encoding, and capitalization to introduce imperceptible variations while preserving readability. **Homograph Substitution** exploits visually similar graphemes, characters, or glyphs with different meanings for imperceptible text modifications. We use VIPER (Eger et al., 2019) and Easy Character Embedding Space (ECES) to obtain optimal homoglyph alternatives. **Formatting Edits** introduces human-invisible disruptions using special escape characters and format-control Unicode symbols to evade detection. We employ newline (\n), carriage return (\r), vertical tab (\v), zero-width space (\u200B), and line tabulation (\u000B) to fragment text at the encoding level while preserving visual coherence. **Case Conversion** alters letter capitalization within a word by converting uppercase letters to lowercase and vice versa. For example, transforming PaSsWoRd into pAsSwOrD disrupts case-sensitive detection while preserving readability.

HMGC (Zhou et al., 2024): A black-box zero-query attack framework. It uses a surrogate model to approximate the detector, ranks words by gradient sensitivity and PPL, and replaces high-importance words via an encoder-based masked language model. Constraints ensure fluency and semantic consistency, while dynamic adversarial learning refines the attack strategy.

Paraphrasing (Wang et al., 2024a): A paragraph-level attack that reorganizes sentence composition to hinder detection. It utilizes Dipper (Krishna

et al., 2024) to reorder, merge, and split multiple sentences, increasing textual variance while preserving meaning.

Code-Switching (Winata et al., 2023): A linguistic modification strategy that substitutes words with their synonyms in different languages. It includes a model-free (MF) approach using a static dictionary (Zhang et al., 2020a; Tiedemann, 2012) for replacements in German, Arabic, or Russian, and a model-required (MR) approach employing the Helsinki-NLP (Helsinki-NLP, 2020) model to translate selected words.

Human Obfuscation: A semantic alteration technique inspired by Semeval Task8 (Siino, 2024), where the initial segment of MGT is replaced with an equally long HWT. The confusion ratio measures the extent of content substitution to increase ambiguity.

Emoji-Cogen (Wang et al., 2024a): A co-generation attack method that inserts emojis into text generation to perturb the output. Emojis are introduced immediately after a token is sampled, before generating the next token, and are removed post-generation, ensuring natural readability while confusing automated detectors.

Typo-Cogen (Wang et al., 2024a): A co-generation attack method that introduces typos during text generation to manipulate lexical structure. Artificial typos are injected into the generated text and subsequently corrected post-generation, preserving overall coherence while disrupting detection models.

In-Context Learning (ICL) (Wang et al., 2024a): A prompt attack method designed to produce human-like outputs that evade detection. It provides the generator with a related HWT as a positive example and a vanilla MGT as a negative example, guiding the model to generate more natural and deceptive text.

Prompt Paraphrasing (Wang et al., 2024a): A prompt attack method rewriting technique that enhances textual variation while maintaining semantic integrity. It utilizes the Pegasus paraphraser (Zhang et al., 2020b) to restructure input prompts.

Character-Substituted Generation (CS-Gen) (Wang et al., 2024a): A prompt attack method that incorporates character substitution strategies within the prompt. The prompt explicitly specifies replacement rules, such as substituting all occurrences of 'e' with 'x' during generation. For example, given the prompt: *"Continue 20 words with all 'e's substituted with 'x's and all 'x's substituted with 'e's: The evening breeze carried a gentle melody..."*, the model generates text following these constraints. A post-processing step then restores the original characters, ensuring a natural final output.

B Experiment on Defense

B.1 Defense Performance under Different Attack Strengths

We evaluate the resistance of defense methods to increasing attack strengths. For text perturbation strategies, we employ four text perturbation strategies: Mixed Edit, Paraphrasing, Code-Switching, and Human Obfuscation in the experiment. Following Wang et al. (2024a), we use Character Edit Distance, BERT Score, BERT Score, and Confusion Ratio as the measure of attack strength for the methods mentioned above, respectively. For adversarial attacks, we choose PWWS TextFooler, BERTAttack, and A2T to attack MGT detectors, and we utilize max query count to quantify the attack strength. In the implementation, we change the limit on the attack strength measures to vary the attack intensity. We show the experimental results in Figure 4.

Our experimental results demonstrate that as the attack strength increases, the ASR on our GREATER-D remains consistently close to zero, whereas other defense methods exhibit significant vulnerabilities under more intensive adversarial attacks. However, it is worth noticing that under Mixed Edit perturbation, GREATER-D performs second-best after EP. This is because EP is originally trained on Mix Edit perturbation and obtains stronger defense against it. The consistent effective defense against varying attack strengths proves the steadiness of our defense method.

B.2 Impact of Synchronous Update of GREATER-D

The synchronous update mechanism plays a crucial role in enhancing the robustness of the target detector. To explicitly demonstrate its contribution, we ablate the generator-side updates and instead use static texts to update the GREATER-D detector. The results are presented in Table 7.

As shown in Table 7, the robustness of the GREATER-D detector significantly degrades when it is trained with static adversarial examples. The ASR converges to approximately 34% after around
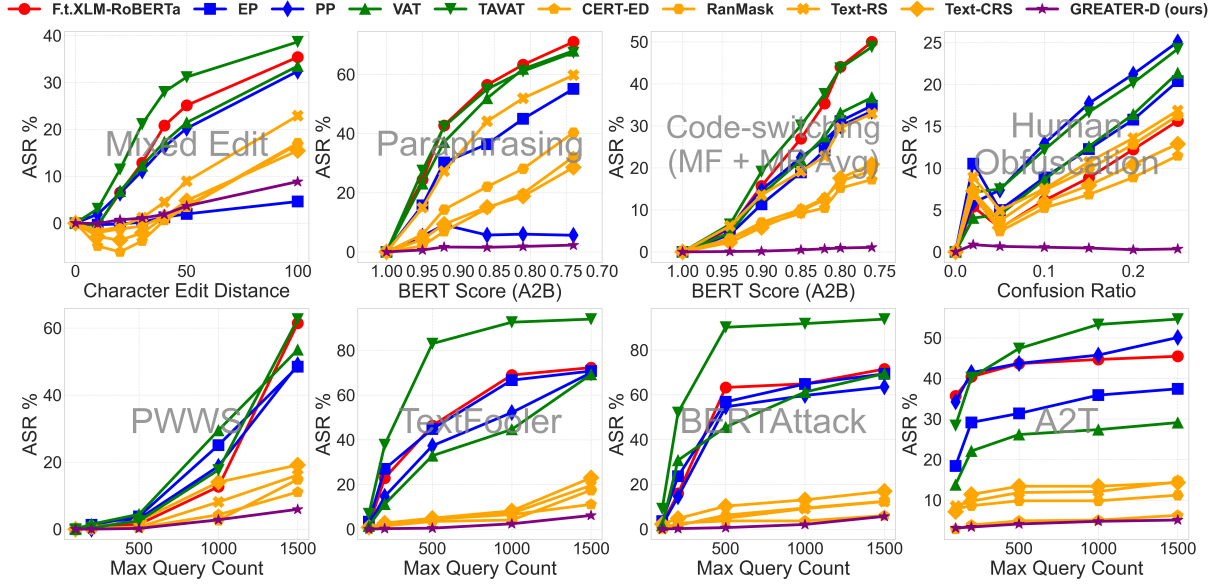
Figure 4: **Defense performance under attack with different strengths.** A lower ASR(%) indicates better defensive performance. A larger character edit distance indicates greater attack intensity in the Mixed Edit Attack. A lower BERT score corresponds to stronger attacks in the Paraphrasing Attack and Code-Switching Attack. A higher obfuscation ratio reflects greater intensity in the Human Obfuscation Attack. Similarly, for PWWS, BERTAttack, TextFooler, and A2T, a larger maximum query count signifies a stronger attack.

Epoch	GREATER-D (w/o synchronous update)	GREATER-D (w/ synchronous update)
1	59.04	57.64
2	52.17	46.08
3	45.52	21.85
4	40.17	10.28
5	34.52	4.77
6	34.08	3.45

Table 7: **Defense performance in ASR (%) of GREATER-D against paraphrasing attacks across different training epochs.**

5 epochs, which is substantially worse than the result achieved with dynamic updates. This demonstrates the critical role of the dynamic update mechanism in enhancing model robustness.

B.3 Defense Performance of Different Epochs

To demonstrate the effectiveness of the adversarial training procedure, we analyze the GREATER-A and GREATER-D performance after each training round. The detailed analysis of training dynamics is in the Table 8. We present the ASR of GREATER-A and defense performance of GREATER-D against Paraphrasing Attack at each epoch from 1 to 6. We find that as training progresses, the GREATER-A becomes stronger at attacking XLM-Roberta detector. Moreover, GREATER-D exhibits more robust defense per-

formance on the attack it is not trained with.

Epoch	GREATER-A ASR ↑	GREATER-D ASR ↓
1	77.04	57.64
2	79.03	46.08
3	84.61	21.85
4	88.94	10.28
5	94.06	4.77
6	96.58	3.45

Table 8: ASR (%) of the Adversary against F.t.XLM-RoBERTa-Base detector and the Detector against Paraphrasing Attack across different epochs.

B.4 Experiments on Multilingual data

We evaluate the multilingual performance of different defense methods using German and Urdu datasets (Siino, 2024), and the results are shown in Table 9. As observed from the table, our model achieves strong performance on both languages and yields the best overall results, indicating its effectiveness in multilingual detection scenarios.

C Evaluations on GREATER-A Performance

C.1 Attack on the real-world detector

Speaking of the fact that the adversary and target model share the same training dataset (but different

Source	Metric	F.t.XLM-RoBERTa	EP	PP	VAT	TAVAT	CERT-ED	RanMask	Text-RS	Text-CRS	GREATER-D
German	*Acc*	68.30	59.84	73.06	67.44	59.37	55.60	61.13	65.04	56.80	72.86
	F1	75.36	71.13	78.40	75.30	71.08	68.90	71.40	73.51	69.55	77.76
Urdu	*Acc*	56.01	55.09	56.88	63.33	59.04	66.60	64.03	70.62	70.21	66.24
	F1	65.27	69.14	69.85	69.07	68.56	74.97	69.81	76.10	76.21	74.89
Overall	*Acc*	62.16	57.47	64.97	65.39	59.21	61.10	62.58	67.83	63.51	69.55
	F1	70.32	70.14	74.13	72.19	69.82	71.94	70.61	74.81	72.88	76.33

Table 9: Performance of different defense methods in a multilingual setting. The best result in each group is highlighted with a green background.

sets), it is necessary to evaluate if the attack conducted by GREATER-A would generalize to other detectors which trained on different datasets. To validate this, we attack a close-sourced commercial detector GPTZero[3] with GREATER-A, which is a strict black-box setting because no other information except for the input and output of the target model is available. The results are in the Table 10. It demonstrates that our GREATER-A remains high ASR compared with other SOTA methods, though we are not aware of the architecture, parameters, or training data at all.

Method	Avg Queries ↓	ASR (%) ↑	Pert. (%) ↓	ΔPPL ↓	USE ↑	Δr ↓
HQA	172.38	63	11.23	133.58	0.8312	66.24
FastTextDodger	162.58	70	10.58	96.25	0.9096	42.15
ABP	133.46	82	9.13	72.78	0.8613	36.18
T-PGD	189.36	42	13.12	168.54	0.8077	60.51
GREATER-A (ours)	42.57	100	7.82	67.13	0.9393	13.39

Table 10: **The attack results of different attack methods on the GPTZero detector.** For each sample, the maximum number of queries is limited to 200. The best result in each group is highlighted with a green background.

C.2 Generalization to Other Detector

To demonstrate the generalization of our GREATER-A, we conduct additional experiments with two zero-shot detectors: Fast-DetectGPT (Bao et al., 2023) and Binoculars (Hans et al., 2024). Moreover, to further demonstrate the generalization ability of GREATER-A, we also test it on a decoder-only detector F.t.GPT2 (Radford et al., 2019) which shares a different architecture with XLM-RoBERTa. As shown in the Table 11, GREATER-A outperforms all comparison methods when attacking zero-shot and decoder-only detectors, which demonstrates the universal effectiveness of GREATER-A.

Method	Fast-DetectGPT	Binoculars	F.t.GPT2
WordNet	51.36	37.35	42.66
Back Translation	37.50	19.13	1.26
Rewrite	10.97	8.88	30.02
T-PGD	56.83	41.91	70.46
HMGC	17.39	31.43	35.08
GREATER-A (ours)	61.14	65.83	74.25

Table 11: **The ASR (%) across different detectors under different attack methods.** The best result in each group is highlighted with a green background.

Method	Avg Queries ↓	ASR (%) ↑	Pert. (%) ↓	ΔPPL ↓	USE ↑	Δr ↓
R+NP	26.61	75.77	11.56	106.29	0.9324	14.41
R+P	53.22	75.77	9.51	58.72	0.9497	12.13
S+NP	31.32	96.58	11.28	85.18	0.9136	21.80
Mask-T	303.82	96.38	8.33	27.12	0.9696	4.73
GREATER-W	33.21	96.38	11.49	65.16	0.9482	10.04
GREATER-WordNet	28.41	80.89	16.68	53.82	0.9199	13.21
GREATER-A	62.63	96.58	7.26	35.22	0.9506	9.21

Table 12: **Ablation study on the GREATER-A.** The best result in each group is highlighted with a green background.

D Ablation Study

To demonstrate the effectiveness of each component in the design of GREATER-D and GREATER-A, we conduct ablation experiments. The ablation models are as follows:

R+NP: Randomly select tokens to perturb and not apply pruning.
R+P: Randomly select tokens to perturb and apply pruning.
S+NP: Select tokens to perturb with the important token identification module but not apply pruning.
Mask-T: Select tokens to perturb with the important token identification but mask them instead of adding perturbation.
GREATER-W: Select words to perturb with the important token identification module instead of tokens.

---

[3] https://gptzero.me/

GREATER-WordNet: Select words to perturb with the important token identification module and substitute them with synonyms.

As shown in Table 12, the original GREATER-A exhibits the most balanced and effective performance in generating adversarial examples. Comparing **R+P** to GREATER-A, we find the ASR drops by **20.81**, indicates that identifying important tokens is of great significance for the attack to be successful. Moreover, greedy pruning is important for maintaining the text quality of adversarial examples. **S+NP** achieves the same ASR with GREATER-A but is left far behind in terms of PPL, USE, and Δr. The perturbation strategy also plays a crucial role in the adversarial attack. Masking the important token instead of perturbing the embedding greatly increases the number of queries. Applying perturbation on the word level or substituting important words with synonyms also degrades the performance of the adversary.

E Mathematical Analysis of Perturbation Rate and Query Complexity

In this section, we provide the theoretical analysis of our proposed adversarial example generation framework, focusing on two crucial metrics: *(i)* the **perturbation rate**, which characterizes the fraction of modified tokens in an adversarial example; and *(ii)* the **query complexity**, which measures the number of queries made to the target detector in a black-box setting. We establish strict upper and lower bounds on both metrics, illustrating the efficiency and effectiveness of our method.

E.1 Perturbation Rate Analysis

Definition 1. *Let Z denote the maximum number of tokens allowed to be modified. Let P be the total number of tokens (out of T) perturbed by* greedy search*, taking integer values in the interval $[1, Z]$, where $1 \leq P \leq Z$. Let Y be the fraction of those P tokens retained (not reverted) by* greedy pruning*, taking real values in $[0, 1]$. Formally,*

$$P \in \{1, 2, \ldots, Z\}, \qquad Y \in [0, 1].$$

We assume P and Y exhibit weak dependence*, meaning they have a small but nonzero covariance $\mathrm{Cov}(P, Y)$.*

Definition 2. *Let $P \cdot Y$ be the expected count of tokens that remain modified. The* perturbation rate ρ *is defined as*

$$\rho = \frac{P \cdot Y}{T},$$

where T is the length of the original text.

Theorem 1. *Let P and Y modeled as truncated normal random variables on $[1, Z]$ and $[0, 1]$, respectively. Then under mild assumptions on the truncation intervals, we have*

$$\mathbb{E}[P] \approx \frac{Z+1}{2} \quad and \quad \mathbb{E}[Y] \approx \frac{1}{2}.$$

Proof. **(1) Truncated Normal for P and Y.** In adversarial text attacks, P often emerges from an aggregation of (approximately) Bernoulli decisions: each of the T tokens has a non-negligible probability of being deemed "important," subject to a global cap of Z. By the Central Limit Theorem (CLT), this sum is close to normally distributed with mean μ_P and variance σ_P^2. Because P cannot exceed Z (and must be at least 1 to induce misclassification), we say

$$P \sim \mathcal{N}_t(\mu_P, \sigma_P^2; 1, Z),$$

where $\mathcal{N}_t(\cdot)$ denotes a normal distribution truncated to the integer range $[1, Z]$. Analogously, once P tokens are selected, *greedy pruning* decides to keep or revert each token, again creating a sum of i.i.d. Bernoulli-like indicators. Dividing by P yields

$$Y = \frac{\text{number of retained tokens}}{P}$$

$$\sim \mathcal{N}_t(\mu_Y, \sigma_Y^2; 0, 1),$$

a (truncated) normal over $[0, 1]$.

(2) Standard Formulas for Truncated Normal Means. From truncated normal theory, if $X \sim \mathcal{N}(\mu, \sigma^2)$ is restricted to $[a, b]$, then its mean is

$$\mathbb{E}[X] = \mu + \sigma \frac{\phi\left(\frac{a-\mu}{\sigma}\right) - \phi\left(\frac{b-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)},$$

where ϕ and Φ are the standard normal PDF and CDF, respectively. Thus, for $P \sim \mathcal{N}_t(\mu_P, \sigma_P^2; 1, Z)$, $\mathbb{E}[P]$ depends on how far μ_P is from the boundaries $\{1, Z\}$. Similarly, $\mathbb{E}[Y]$ depends on truncation at $[0, 1]$.

(3) Approximate Symmetry in Practice. For symmetric truncated normal distributions P and Y, it is evident that the following expectations hold:

$$\mu_P = \frac{Z+1}{2}, \quad \mu_Y = \frac{1}{2}.$$

So long as μ_P and σ_P ensure that $\frac{1-\mu_P}{\sigma_P}$ and $\frac{Z-\mu_P}{\sigma_P}$ are not extreme, the truncation does not drastically

shift the mean from μ_P. Concretely, if Z is sufficiently large and $\mu_P \approx \frac{Z+1}{2}$, then

$$\mathbb{E}[P] \approx \mu_P \approx \frac{Z+1}{2},$$

Likewise, if $\mu_Y \approx \frac{1}{2}$ and σ_Y is moderate, $\mathbb{E}[Y] \approx \frac{1}{2}$ despite the boundaries $[0, 1]$.

Combining (1), (2), and (3), Theorem 1 is proved.

$\square$

Theorem 2. *Let T be the length of the original text and M be the maximum number of tokens that can be perturbed. Then, the perturbation rate ρ satisfies $\frac{1}{T} \leq \rho \leq \frac{Z}{T}$, and its expected value is approximately $\mathbb{E}[\rho] \approx \frac{Z}{4T}$.*

Proof. **(1) Basic Bounds.** Since P is at least 1 and at most M, and Y is at least 0 and at most 1, the product $P \cdot Y$ satisfies

$$1 \leq P \leq Z, \qquad 0 \leq Y \leq 1$$

$$\implies 1 \leq P \cdot Y \leq Z.$$

Dividing by T yields the strict bounds

$$\frac{1}{T} \leq \frac{P \cdot Y}{T} \leq \frac{Z}{T}.$$

The lower limit $1/T$ reflects that at least one token must change to induce a misclassification; the upper limit $\frac{Z}{T}$ follows from the maximal Z token modifications.

(2) Expected Product with Weak Dependence. By definition, the covariance between P and Y is

$$\mathbb{E}[PY] = \mathbb{E}[P]\,\mathbb{E}[Y] + \mathrm{Cov}(P, Y).$$

When the detector is relatively robust, *greedy search* skews P toward larger values (close to M), and pruning skews Y toward retention (close to 1). In that case, $\mathbb{E}[P]$ is large, $\mathbb{E}[Y]$ is near 1, and a small positive covariance implies

$$\mathbb{E}[P \cdot Y] > \mathbb{E}[P]\,\mathbb{E}[Y].$$

However, the final mean number of changed tokens cannot exceed M. Conversely, if the detector is weak, $\mathbb{E}[P]$ may approach 1, and $\mathbb{E}[Y]$ might be relatively modest, implying

$$\mathbb{E}[PY] < \mathbb{E}[P]\,\mathbb{E}[Y] + |\mathrm{Cov}(P, Y)|.$$

In all cases, the weak dependence ensures $\mathrm{Cov}(P, Y)$ is bounded such that

$$1 \leq P \cdot Y \leq Z,$$

preventing the expected perturbation rate from falling below $\frac{1}{T}$ or above $\frac{Z}{T}$.

(3) Characteristic Mean $\frac{Z}{4T}$. By Theorem 1, we have $\mathbb{E}[P] \approx \frac{Z+1}{2}$ and $\mathbb{E}[Y] \approx \frac{1}{2}$. If $\mathrm{Cov}(P, Y)$ is small or near zero, then

$$\mathbb{E}[P \cdot Y] = \mathbb{E}[P]\,\mathbb{E}[Y] + \mathrm{Cov}(P, Y)$$

$$\approx \frac{Z+1}{2} \times \frac{1}{2} = \frac{Z+1}{4}.$$

For large Z, $\frac{Z+1}{4T} \approx \frac{Z}{4T}$. If $\mathrm{Cov}(P, Y)$ modestly raises or lowers this sum, the final mean $\mathbb{E}[PY]$ still cannot breach the fundamental $[1, Z]$ interval. This shows that $\frac{Z}{4T}$ is a natural approximate pivot for the average perturbation rate under moderate parameters.

In Section 5.2, we set $Z = 0.3T$. If the number of iterations exceeds this upper bound, the attack is considered a failure. Therefore, the expected perturbation rate is

$$\frac{0.3T}{4T} \approx 7.5\%,$$

which is very close to the result obtained in the Section 5.2 (7.26%).

Combining (1), (2), and (3), Theorem 2 is proved.

$\square$

Note that if the detector forces *greedy search* to repeatedly fail early (producing a right-skewed P-distribution concentrated near Z) and *greedy pruning* is left-skewed (so that $\mathbb{E}[Y]$ is close to 1), then the mean $\mathbb{E}[P \cdot Y]$ exceeds $\frac{Z+1}{4}$, but still cannot exceed Z. Conversely, if *greedy search* finds success quickly, giving a left-skewed P-distribution near 1, then $\mathbb{E}[P]$ might drop below $\frac{Z+1}{2}$ but never below 1; similarly, if *pruning* is so aggressive that $\mathbb{E}[Y]$ is significantly below $\frac{1}{2}$, the mean also decreases. Consequently, the expected perturbation rate remains strictly within the $\left[\frac{1}{T}, \frac{Z}{T}\right]$ interval, but its exact value depends on the interplay of these skewed distributions. For moderate skewness on both P and Y, $\frac{Z}{4T}$ is a characteristic outcome.

E.2 Query Complexity Analysis

Definition 3. *Let Q_{G} be the total number of queries made by* greedy search *and Q_{P} be the total number made by* greedy pruning. *Define the overall* query complexity:

$$Q = Q_{\mathrm{G}} + Q_{\mathrm{P}}.$$

Theorem 3. *Let T be the length of the text, and let $Z = 0.3\,T$ be the maximum iteration count for* both *greedy search* and *greedy pruning. Then the total number of queries Q used to construct one adversarial example lies within $1 \leq Q \leq 2Z$, which implies $Q = O(T)$ and $Q = \Omega(1)$. Moreover, if the* average *perturbation rate is relatively low, then the number of required iterations typically shrinks, resulting in a correspondingly smaller Q.*

Proof. **(1) Basic Range of Queries.** The minimal number of queries is 1, occurring if *greedy search* succeeds in its very first attempt, requiring no *greedy pruning*. Conversely, if both *greedy search* and *greedy pruning* use their maximum of Z single-query iterations, we have

$$Q = Q_{\mathrm{G}} + Q_{\mathrm{P}} \ \leq\ Z + Z \ =\ 2\,Z.$$

Since $Z = 0.3\,T$, $Q \leq 2\,Z$ translates to $Q = O(T)$. The trivial lower bound of 1 implies $Q = \Omega(1)$.

(2) Perturbation Rate and Query Trade-Off. Let ρ denote the fraction of tokens altered in a final adversarial example, as described in the perturbation rate analysis. A *smaller* ρ typically indicates that the detector is fooled by changing fewer tokens, implying fewer iterative steps in *greedy search*. Thus, a lower ρ correlates with *fewer* queries: once the necessary (small) set of tokens is found, misclassification is often achieved without exhausting all Z iterations. On the other hand, a higher ρ suggests more alterations, potentially requiring more query rounds to finalize a successful attack.

(3) Expected Complexity under Moderate Perturbation. From the previous analysis, the average perturbation rate can hover around $\frac{Z}{4T}$ under typical conditions. This moderate ρ implies that *greedy search* rarely needs the full Z steps to identify the required modifications, and *greedy pruning* likewise terminates without iterating over all Z. Consequently, the *expected* Q is substantially below the worst-case $2\,Z$. Formally, we have

$$\mathbb{E}[\,Q_{\mathrm{G}}\,] \ <\ Z, \quad \mathbb{E}[\,Q_{\mathrm{P}}\,] \ <\ Z,$$

$$\implies \quad \mathbb{E}[\,Q\,] \ =\ \mathbb{E}[\,Q_{\mathrm{G}}\,] + \mathbb{E}[\,Q_{\mathrm{P}}\,] \ <\ 2\,Z.$$

thus still preserving $O(T)$ upper complexity while being strictly smaller on average.

Combining (1), (2), and (3), Theorem 3 is proved.
$\square$

F   Adversarial Training Process

Algorithm 3 shows the detailed optimization process of GREATER-A and GREATER-D. The two components are updated alternatively in the same training step.

Algorithm 3 Adversarial Training Procedure

1: **Input:** Training set D_{train}, surrogate detector $\mathcal{M}_{sur}$.
   **** *training phase begins* ****
2: **Initialize:** target detector $\mathcal{M}_{tar}$, importance scoring network $\mathcal{F}_\theta$ and $epoch \leftarrow 0$.
3: **while** $epoch < epoch_{max}$ **do**
4: **for** each batch samples $\{X_i, c_i\}_{i=0}^N$ in D_{train} **do**
5:   $D_{adv} \leftarrow \{\}$
6:   **for** each MGT in $\{X_i, c_i\}_{i=0}^N$ **do**
7:    Obtain the last layer hidden state by Eq.(1).
8:    Obtain the importance score by Eq.(2).
9:    Construct the Important-token Set **I** by Eq.(3).
10:    Execute the Greedy Search Algorithm1.
11:    Execute the Greedy Pruning Algorithm2.
12:    Get the adversarial example:
     $D_{adv} \leftarrow D_{adv} \cup \{X_i, c_i\}_{i=0}^N$.
13:   **end for**
14:   $D_{adv} \leftarrow D_{adv} \cup \{\tilde{X}_i, c_i\}$.
15:   Classify D_{adv} via $\mathcal{M}_{tar}$ and obtain the output.
16:   Calculate loss $\mathcal{L}_{\mathrm{A}}$ by Eq.(11).
17:   Calculate loss $\mathcal{L}_{\mathrm{D}}$ by Eq.(12).
18:   Update $\mathcal{M}_{sar}$ and $\mathcal{F}_\theta$ via SGD (1951).
19: **end for**
20: **end while**
21: **Output:** Trained detector $\mathcal{M}_{tar}$ and $\mathcal{F}_\theta$.

G   Case Study

Table 13 presents a case study of our adversary GREATER-A, in which our approach GREATER-A outperforms other SOTA methods regarding semantic preservation and reduction in perturbation rate.

Method	Text	Result	Pert. (%)	Queries
Original MGT	Suggested by the Scottish Parliamentary Constituencies Commission in 2003, and adopted at Holyrood on 1 May 2004. Area of Scotland's 32nd largest council area - covering parts of East Renfrewshire Council Area	Machine-written 😖	-	-
PWWS	Suggested by the Scottish Parliamentary Constituencies Commission in 2003, and adoptions at Holyrood on 1 Probability 2004. Area of Scotland's 32nd highest council area - covering item of East Renfrewshire Council Area	Human-written (Succeeded) 😊	11.76	107
TextFooler	Suggested by the Scottish Parliamentary Constituencies Commission in 2003, and adopted at Holyrood on 1 May 2004. Area of Scotland's 32nd largest council area - covering parts of East Renfrewshire Council Area	Machine-written (Failed) 😕	-	-
BERTAttack	Suggested by the Scottish rural Constituencies Commission in 2003, and abolished at Holyrood on 1 May 2004. Area of ward 32nd most council constituency - almost all of East Renfrewshire Council Area	Human-written (Succeeded) 😊	20.59	60
A2T	Suggested by the Scottish Parliamentary Constituencies Commission in 2003, and adopted at Holyrood on 1 May 2004. Area of Scotland's 32nd largest council area - covering parts of East Renfrewshire Council Area	Machine-written (Failed) 😕	-	-
FastTextDodger	Suggested by the Scottish Parliamentary Constituencies Commission in 2003: and adopted at Holyrood on 1 May 2004. Area of' 's 32nd largest council area - covering parts of East council Council	Human-written (Succeeded) 😊	14.71	73
ABP	Suggested aside the Scottish Parliamentary Constituencies Commission indium 2003, and adopted atomic number 85 Holyrood on ace Crataegus oxycantha 2004. Area of Scotland's 32nd largest council area - covering parts of East Renfrewshire Council Area	Human-written (Succeeded) 😊	23.53	71
HQA	Suggested by the Scottish Parliamentary Constituencies Commission in 2003, and adopted at Holyrood on 1 May 2004. Area of Scotland's 2004. sphere council area thirty-second prominent council of East Renfrewshire Council Area	Human-written (Succeeded) 😊	20.59	44
T-PGD	Introduced. by. Scottish Parliamentary Resituiances Commission in 2002, and adopted at Holyrood on 1 May 2004. All of Scotland's 32 The largest councils area - covering parts of East Renfrewshire Council Area	Human-written (Succeeded) 😊	26.47	56
Greater-A	Suggested by the Grave Parliamentary Constituencies Commission in 2003, and adopted at Holyrood on 1 May 2004. Area Scotland 32nd biggest council area - covering parts of East Renfrewshire Council Area	Human-written (Succeeded) 😊	5.88	12

Table 13: Case study of semantic preservation of the adversarial texts generated by various attack methods. Words modified during the attacks are highlighted in red.