Stumbling Blocks: Stress Testing the Robustness of Machine-Generated Text Detectors Under Attacks

Anonymous ACL submission

Abstract

The widespread use of large language models (LLMs) is increasing the demand for methods that detect machine-generated text to prevent misuse. The goal of our study is to stress test the detectors' robustness to malicious attacks under realistic scenarios. We comprehensively study the robustness of popular machine-generated text detectors under attacks from diverse categories: *editing*, *paraphrasing*, *prompting*, and *co-generating*. Our attacks assume limited access to the generator LLMs, and we compare the performance of detectors on different attacks under different budget levels. Our experiments reveal that almost *none* of the existing detectors remain robust under all the attacks, and all detectors exhibit different loopholes. Averaging all detectors, the performance drops by 35% across all attacks.[1]

1 Introduction

LLMs are becoming increasingly adopted in information seeking scenarios, assistive writing, translation, mental health support, and many more (Zhao et al., 2023a). Their evolving capabilities to generate human-like and persuasive language raise wide concerns about misuse, e.g., deception, academic misconduct, and disinformation (Zellers et al., 2019; Weidinger et al., 2021; Kumar et al., 2022; Feng et al., 2024), and it becomes harder for humans to distinguish machine-generated texts (MGT) from human-written texts (HWT) (Dugan et al., 2023). As a result, much recent work focus on automatic MGT detection to mitigate the risks (Liu et al., 2022; Mitchell et al., 2023; Kirchenbauer et al., 2023a; Mao et al., 2024) (related work discussed in Appendix A).

In this work, we focus on potential malicious attacks that attempt to deceive the detector using various attack strategies. Existing works on this topic mostly focus on the robustness of specific detectors or particular attack methods. For example, Liu et al. (2022) specifically evaluate the token editing attack for model-based detectors, and Zhang et al. (2023) assay the topic-shifting attack for metric-based detectors, etc. To the best of our knowledge, in the literature, there is no thorough comparative evaluation of robustness of machine-generated text detectors against malicious attacks, covering a wide range of detectors and attacks.

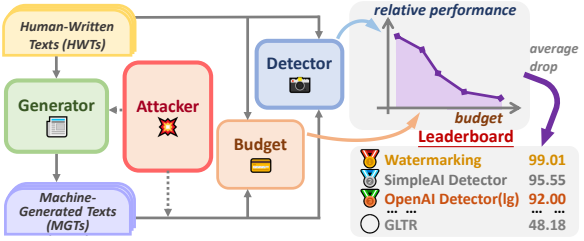

Figure 1: **Pipeline of the study.** The attacks are carried out on the machine-generated texts before, during, or after generation. Each attack is applied with different perturbation levels, denoted as budgets (§4).

With this goal, we study the robustness of 8 prevalent MGT detectors from 3 categories under 12 realistic attacks (§6, Table 1), including editing, paraphrasing, prompting, co-generating, etc. The majority of the attacks in this paper are proposed or attempted for the first time. For a fair comparison across detectors and attacks, we utilize a series of metrics to measure the perturbation level of each attack, which we term "budget" (§4). Strikingly, our experiments (§6.1) reveal that **almost none of the existing detectors remains robust under all the attacks**, showing a variety of potential weaknesses or loopholes. For example, about 2 to 6-character editing by typo insertion can severely deceive metric-based detectors, such as DetectGPT (Mitchell et al., 2023), to perform worse than a random prediction (§6.2), etc. Hence, we view the attacks as the *stumbling blocks* for current MGT detectors toward robustness. Moreover, we interpret the reasons behind the detectors' weaknesses under

[1] Code and data will be released in the public version.

Attack Category	Method	Model-Free?	Level	Access	Detailed Descriptions
Editing (§6.2) `post-generation`	Typo Insertion	✓	Character	None	Create typos by inserting, deleting, substituting, and transposing mainly.
	Homoglyph Alteration	✓	Character	None	Change English characters into visually similar Unicodes, e.g., Cyrillic characters.
	Format Character Editing	✓	Character	None	Change or insert formatting characters, including zero-width whitespace `\u200B` insertion, and shift character editing, e.g., `\n`, `\r`, `\u000B` (vertical tab), etc.
Paraphrasing (§6.3) `post-generation`	Synonyms Substitution	opt ✓ or ✗	Word	None	For model-free (✓) setting, retrieve a synonym from a static dictionary; for model-based (✗) setting, utilize a LLM to generate synonyms list given context.
	Span Perturbation	✗	Span	None	Use a masked LM (Raffel et al., 2020) to rewrite spans of tokens by masked filling.
	Inner-Sentence Paraphrase	✗	Inner-Sent.	None	Use Pegasus (Zhang et al., 2020) to paraphrase each sentence of the text and then join them.
	Inter-Sentence Paraphrase	✗	Inter-Sent.	None	Paraphrase with Dipper (Krishna et al., 2023), a paragraph-level paraphraser that can re-order, split, and merge sentences meanwhile paraphrasing each sentence.
Prompting (§6.5) `pre-generation`	Prompt Paraphrasing	✗	Inter-Sent.	Prompting	Paraphrase the raw prompt before generation using Pegasus.
	In-Context Learning	✗	Inter-Sent.	Prompting	Given the example of HWT and MGT as positive and negative demonstrations when generating MGT on the same prompt.
	Character-Substituted Generation	✗	Inter-Sent.	Prompting	Prompt to ask the model to generate the text with specific character substitution criteria and recover the output after finishing the whole generation.
Co-Generating (§6.4) `on-generation`	Emoji Co-Generation	✓	Inter-Sent.	Decoding	Compulsorily generate or insert an emoji after finishing each sentence while recurrent generation and remove all the emojis after finishing the whole text.
	Typo Co-Generation	✓	Inter-Sent.	Decoding	Preset character substitution rules and execute the rules when finishing sampling each token and recover them after finishing the whole text generation.

Table 1: **Overview of the attacks.** 'Model-Free' means whether the attacker is free from using any additional language model or not. 'Access' indicates the access to the generator needed when doing the attack (details in §6 and examples in Table 15).

attacks, and we further introduce out-of-the-box patches with inferior performance in some scenarios (further defense discussed in Appendix E.1).

We build a robustness leaderboard (Table 2, and the pipeline is illustrated in Figure 1) by averaging results from different attacks. We find that **watermarking** (Kirchenbauer et al., 2023a) **performs best for robust MGT detection to its applicable attacks.**[2] Next, **model-based detectors are more robust than metric-based ones in most cases.** Overall, this study aims to raise awareness of the detection vulnerabilities and the urgency of more robust methodologies, thereby turning the *stumbling blocks* into *stepping stones*.

2 Problem Formulation

Threat Model. Figure 1 shows the overall pipeline. There are three roles in the problem: *generator* (§3), *detector* (§3), and *attacker* (Table 1, §6). The task for the detector is to classify whether a given piece of text is human-written (HWT) or machine-generated (MGT) from the generator LM. In the attacked scenario, before the MGT is sent to the detector, an attacker could tamper with the text or the generator, attempting to deceive the detector into classifying the MGT as HWT. We compute the *budget* (§4) of each attack to measure its impact on text quality and semantics.

Scope. For a realistic scenario, we set the scope of our robustness evaluation under attack as follows: ***i***) We assume that the attacker does **not** have any knowledge or access to the **detectors**. ***ii***) The attacker only has **limited** access to the **generators**: We assume to have prompting access with tunable sampling hyper-parameters for the following reason: currently, most top-performing LLMs accessible to users are closed-source (e.g., GPT-4, Claude), to which we only have API access or a panel including a prompt input and sampling settings (OpenAI, 2022a). Due to the same reason, adversarial attacks (Li et al., 2018; Le et al., 2022) are not covered in this study. ***iii***) For a holistic comparison, we apply each attack on different perturbation levels (e.g., number of typos), termed as budgets (§4).

3 Generators and Detectors

We select GPT-2 XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), and LlaMA-2 (7B-hf) (Touvron et al., 2023a) as the representative open-source generators, and Text-

[2]Watermarking requires logit-level access to the generator model and has the risk of negatively impacting text quality.

Davinci-003 (OpenAI, 2022b) and GPT-4 (OpenAI, 2023) as the closed-source generator representatives. **All the generators shared similar results under attacks** (Appendix G.3). We select GPT-J (6B) as the default generator to show the results in §6 if unspecified (we empirically find stronger generative LMs are not as well good at detecting when used for metric-based detectors). The results of LlaMA-2 and GPT-4 will be additionally shown in Appendix G.3 and §6. For closed-source generators, some of the detectors can not be applied due to the requirement of white-box parameters.

Current MGT detectors could be classified into 3 high-level categories, as we introduce below. We include representative detectors from each category for our evaluation. A detailed introduction of the detectors is deferred to Appendix C.2.1.

Metric-Based Detector relies on the inferenced log-probability from the generator LLM, and adopts a threshold for classification.[3] Detectors for this type does not require any training. We include *GLTR* (Gehrmann et al., 2019; Solaiman et al., 2019), *Rank* and *LogRank* (Solaiman et al., 2019), and *DetectGPT* (Mitchell et al., 2023) as representative approaches in the category.

Fine-Tuned Detector is trained on a pretrained language model (PLM) in a supervised method with a classification loss. We include *OpenAI Detector* (Solaiman et al., 2019), *SimpleAI Detector* (Guo et al., 2023), and *Fine-tuned DeBERTa* as representative models in the category.

Watermark-Based Detector adds algorithmically detectable signatures into texts during generation. Kirchenbauer et al. (2023a) is a representative approach, which adds a token-level bias in the decoding stage (represented as *Watermark* afterward).

We follow the recommended configurations for most detectors. Detailed hyperparameters are included in the Appendix C.2.2.

4 Budget of Attacks

As stated in §2, to measure the perturbation level of attacks on the generated texts, we utilize a series of text generation evaluation metrics as the budget of attacks, covering syntactic- or semantic-level perturbation. A strong attack should induce large detection performance degradation with a relatively small budget.

For the editing attacks, we use *Levenshtein Edit Distance* (Levenshtein, 1965) as the major budget, which is the minimum number of single-character edits, including insertions, deletions, and substitutions. A larger distance represents a larger attack budget. Additionally, we also record *Jaro Similarity* (Jaro, 1989).[4]

To measure the quality of texts under the attacks that change the semantic meaning (e.g., prompting attacks and co-generating attacks), we utilize *Perplexity* under LlaMA-7B-hf (Touvron et al., 2023b) and *MAUVE* (Pillutla et al., 2021). We use MAUVE to compare the distribution gap between MGTs and HWTs. MGTs are used to estimate the model distribution, and HWTs are used to estimate the target distribution (the setting is abbreviated as 'M2H'). Lower Perplexity or higher MAUVE (M2H) represents better quality and a smaller budget. Table 6 shows the unattacked value for reference.

For the attacks that do not change semantics meaning, e.g., paraphrasing, we use *BERTScore* (Zhang et al., 2019) as the major metric for the budget. We utilize it to compare the similarity between MGTs after the attack to MGTs before the attack. In this scenario, attacked MGTs are the candidates for BERTScore, while unattacked MGTs are the reference (the setting is abbreviated as 'A2B'). The BERTScore we used is rescaled. A larger BERTScore (A2B) value means a smaller budget in the attack. Besides, we also record *BARTScore* (Yuan et al., 2021) and *Cosine Similarity*, which shows equivalent results.

See Table 16 for more details on the metrics for the attack budget. Appendix G.1 show the correlation among all metrics and they show highly similar trends of attacked performance.

5 Experiment Setting

Data Setting. Following the setting of Pu et al. (2023), we generate News-style texts with a proper sampling strategy for each generator, detailed in Appendix C.1. Our study can be readily applied to data from other domains. The prompts used for MGT generation are the first 20 tokens of HWTs in the dataset. The setting of the sampling strategy aims to prevent repetition, measuring by duplicate n-grams (Welleck et al., 2019). The size of the

[3]The setting of threshold largely impacts the detection accuracy, but it is out-of-the-scope of this paper's focus. Thus, we mainly use threshold-free metrics (e.g., *AUC ROC* and *TPR@FPR*) in experiments (detailed in §5).

[4]The edit distance, Jaco similarity, and cosine similarity are implemented based on the `string2string` (Suzgun et al., 2023) package.

Leaderboard: MGT Detector Robustness					
Detector	*Edit*	*Para.*	*Prompt*	*CoGen.*	***Avg.***
Watermark	**99.86**	**97.17**	- -	**99.99**	**99.01**
SimpleAI Det.	**108.1**	97.51	81.58	95.04	**95.55**
OpenAI Det.-Lg	57.77	**97.84**	**105.2**	**107.2**	92.00
Model. Avg.	76.65	92.08	97.57	92.22	89.63
F.t. DeBERTa	104.1	81.49	99.09	64.28	87.24
OpenAI Det.-Bs	36.63	91.46	104.4	102.4	83.71
DetectGPT-1d	**74.82**	**75.32**	**102.8**	**66.46**	**79.85**
DetectGPT-10d	62.67	64.40	97.68	49.78	68.63
DetectGPT-10z	56.41	59.73	93.88	43.08	63.28
Metric. Avg.	51.82	61.89	91.26	33.49	59.62
LogRank	41.76	58.38	84.44	11.20	48.95
Rank	36.46	57.68	81.00	20.08	48.81
GLTR	38.82	55.80	87.79	10.32	48.18

Table 2: **The overall robustness leaderboard of MGT detectors** by averaging the relative AUC ROC percentage across all attack budget levels in §6, ranking downwards by the overall average. '*Metric. Avg.*' and '*Model. Avg.*' represent the average performance of metric-based and model-based detectors. Bolding indicates the best performance in each detector category, and worse performance with drops larger than 70% are in orange.

Absolute MGT Detector Performance w/o Attack					
Detector	*AUC*	*TF=5*	*TF=10*	*TF=20*	*ACC*
GLTR	84.46	39.00	53.40	71.60	76.00
Rank	68.13	22.60	35.60	46.80	63.60
LogRank	87.36	50.00	65.60	78.20	79.00
Entropy	51.84	7.60	14.60	26.40	50.80
DetectGPT-1d	68.66	15.80	27.40	45.80	62.10
DetectGPT-10d	83.12	21.60	43.80	71.20	75.80
DetectGPT-10z	85.16	30.80	50.80	73.20	76.20
OpenAI Det.-Bs	83.12	42.40	56.20	69.00	75.00
OpenAI Det.-Lg	88.55	53.60	65.60	78.00	79.00
SimpleAI Det.	87.98	81.20	82.60	84.60	84.40
F.t. DeBERTa	91.90	5.40	49.20	99.60	88.80
Watermark	99.94	99.80	99.80	99.80	99.99

Table 3: **The performance of the detectors in the unattacked scenario (absolute value).** For short, 'AUC' is ROC AUC, 'TF=5' is TPR@FPR=5%, 'ACC' is Accuracy, 'Det.' is Detector, and 'F.t.' is Fine-tuned.

training, evaluation, and testing set is 8,000, 1,000, and 1,000, respectively, with balanced labels.

Metrics for Detector Performance. The metrics we use to evaluate detection performance are binary classification metrics *AUC ROC* and *TPR@FPR*. AUC ROC is the area under the receiver operating characteristic curve. TPR@FPR is the true positive rate when the false positive rate is at a specific percentage. Under our setting, it is equivalent to *Attack Success Rate (ASR)* (Tsai et al., 2019). We mainly show *TPR@FPR=5%*, and TPR@FPR=10% and =20% are additionally recorded in the Appendix G.2. We do not involve *Accuracy* and *F1-score* because those metrics are dependent on the setting of the threshold for metric-based detectors, which could be biased in the comparison. Notably, we report all the metrics of attacked scenarios in relative value to the unattacked performance (Table 3) for clearer comparison.

6 Attacks and Results

In this section, we describe the attack methodologies and results divided by attack category. We view the degraded performance under attacks of various detectors as *stumbling blocks* to robust MGT detection. Further, we analyze the defects and propose defense patches in each category to explore the potential of *turning stumbling blocks into stepping stones*. Table 1 is an overview of all attacks and Table 15 shows some examples.

6.1 Overall Message

For readers who want a high-level overview of our findings, we show the overall results and messages ahead here by aggregating results from all types of attacks covered in our work. We will introduce and discuss the detailed attacks and results in the following subsections (§6.2 - §6.4).

Leaderboard. Overall, we build a leaderboard of detector robustness averaging all the performance datapoints under attacks. The relative AUC ROC under attack[6] are as shown in Table 2. A high relative AUC ROC means that the detector is robust to the attack. According to the leaderboard, **watermarking is most robust to accessible attacks**[7]. Following, SimpleAI Detector and OpenAI Detector (large) rank second and third. Moreover, **model-based detectors are more robust than metric-based detectors** in most cases. Additionally, we report the absolute performance of the detectors without attacks in Table 3, which should also be considered while selecting suitable detectors.

Detector Defect Review. We summarize the de-

[5]The x-ticks in format (ZWS) character editing is twice the ones in typo and homoglyph because the Unicode is 2 bytes when computing edit distance.

[6]'Relative AUC ROC under attack' is the percentage of the AUC ROC in attacked scenarios divided by the unattacked AUC ROC, to show the relative performance drop of the detectors under attack. Detailed in §5.

[7]Some prompting attacks can not be done on watermark since in need of white-box models and compatibility to the watermarking decoding.

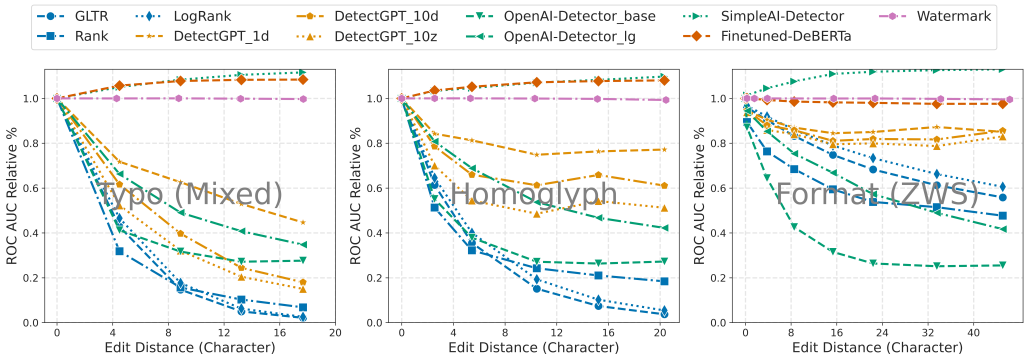

Figure 2: **Performance drop of the detectors under the editing attacks.** We show the mixed setting for typo insertion, and the zero-width whitespace setting (ZWS for short) for format character editing. The budget on the x-axis is the edit distance at character level (↑ a larger number represents a stronger attack). The color of dashed lines indicates the category of detectors.[5]

fect for each detector as follows. GLTR, Rank, and LogRank have an average performance drop of 51.35% under all attacks, especially not robust to editing, paraphrasing, and co-generating attacks. In comparison DetectGPT shows better robustness (average 29.41% drop), especially on paraphrasing and co-generating attacks. Among fine-tuned detectors, the strongest attack method varies. SimpleAI Detector drops performance on paraphrasing and prompting attacks, OpenAI Detectors drop on editing, and F.t. DeBERTa performs worse on co-generating while it keeps decent robust to other attacks. We empirically find a larger model size of OpenAI Detectors eases the robustness drawback. Notably, watermarking is robust to all applicable attacks, but it requires decoding-time access to the generator compared with other detectors.

6.2 Editing Attacks

The first attack type we explore is the editing attacks, which are applied to the generated texts by minor editing at the character level without any change in semantics at the post-generation stage. Thus, editing attacks are at a low granularity. Some of the attacks might cause the text to lose minor quality and readability. Below, we will introduce three attack types.

6.2.1 Approaches

Typo Insertion intentionally adds a few typos into generated texts. We consider four main kinds of typos in English keystroke scenarios: insertion, deletion, substitution, and transposition (Kukich, 1992). Aside from testing on each kind, we propose a mixed typo insertion to mimic the realistic scenario according to the distribution investigated by Baba and Suzuki (2012).[8] Also, we additionally take letter frequency into account when selecting the characters to be attacked (Pavel, 2000).

Homoglyph Alteration uses graphemes, characters, or glyphs with visually identical or very similar shapes but different meanings for imperceptible replacements, first introduced in the cyber security domain (Gabrilovich and Gontmakher, 2002). We use VIPER (Eger et al., 2019) (Visual Perturb) Easy Character Embedding Space (ECES) to get the best homoglyph alternative of the selected character.

Format Character Editing, also named Discreet Alteration (Kirchenbauer et al., 2023a), uses special escape characters and format-control Unicodes as human-invisible disruptions to deceive detectors. See details in Appendix F.2.

We do all the editing on the character level, and the budget is measured by edit distance. Also, we do at most one edit per word.

6.2.2 Results and Analysis

As shown in Figure 2, all metric-based detectors and some fine-tuned ones drop dramatically, while only SimpleAI-Detector and Fine-tuned DeBERTa maintain good performance. Specifically, **around 2 to 6 characters editing of typos or homoglyph per text can degrade the performance of most detectors to be worse than random** (The average length of texts is around 120 tokens). All metric-based methods show a continuous decrease while attack budgets increase. In typo and homoglyph attacks, the decrease can mount up to a total failure with ROC AUC near 0. In comparison, the DetectGPT is more robust than the others. e.g., its drop converges to about 0.5 under homoglyph alteration

[8] substitution 55.6%, insertion 20.3%, transposition 1.1%, deletion 23.0%.

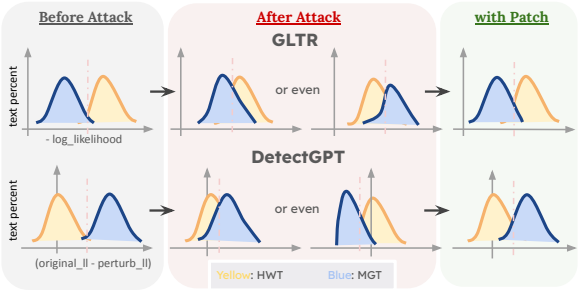

Figure 3: Illustration of the distribution of the metric value of the metric-based detectors before the attack, after the attack, and after patching (an out-of-the-box defense we proposed in §6.2.3). The light-red dotted lines are the optimal decision boundaries.

while GLTR and Log Rank are near 0. But, DetectGPT with fewer perturbed samples (`_1d`) is more robust than larger ones (`_10d` and `_10z`), which is counterintuitive.

Among fine-tuned detectors, OpenAI Detectors show a similar unsoundness as metric-based ones. And their drops are most significant in format character editing. SimpleAI Detector and F.t.-DeBERTa show great robustness to all editing attacks. Moreover, comparing two underperforming OpenAI Detectors of different sizes indicates that larger classification models could be more robust than smaller ones under editing. The watermarking method is also very robust to attacks, i.e., keeping the AUC ROC above 99%.

In addition, all four individual types of typo share similar negative impacts as the mixed version, as shown in Appendix F.1. Also, we observe similar drops for all format character editing attacks, while zero-width whitespace is more effective.

Interpretation. The assumption of metric-based detectors is that HWTs have smaller log-probabilities than MGTs when inferenced by the generator model. However, editing attacks can effectively decrease the next-token probabilities, leading to indistinguishable situations of the distribution curves and even inverse relative relationships, which cause completely wrong predictions (ROC AUC near 0) as the budget increases. Figure 3 shows a detailed illustration, taking GLTR and DetectGPT as examples. In the figure, after the attack, a larger overlap of the two curves (column 2) means more severe indistinguishability, and the interchange of the relative positions of the two curves (column 3) leads to wrong predictions. From this intuition, we attempt to patch the issue in §6.2.3 by removing anomalies. Discuss for fine-tuned and watermarked detectors are deferred to Appendix D.1.

Detector	Before Att.	After Att.	w/ Patch
DetectGPT-1d	0.6866	0.4299	0.5111
DetectGPT-10d	0.8312	0.3301	0.6048
DetectGPT-10z	0.8516	0.2735	0.6032

Table 4: Performance of DetectGPT after patching under typo insertion attack in terms of AUC ROC.

6.2.3 Out-of-the-box Defense Patch

In this section, we propose a simple patch for the under-performing DetectGPT approach. As the editing attacks mainly cause extremely low token probabilities to deceive the classification, we view them as anomaly points to filter them out. Specifically, for each text, the top k% tokens with the lowest probabilities would be prevented from being masked and perturbed when doing mask-filling. Next, we do not take their token probability into the computation. Table 4 show the patch recovers performance by 0.2285 on average for 3 settings.

Other potential patches include adversarial training (Goodfellow et al., 2014), visual character embeddings (Wehrmann et al., 2019) for homoglyph, and preprocessing with grammatical error correction (Bryant et al., 2022). These approaches are more costly, and we leave them to future work.

6.3 Paraphrasing Attacks

Paraphrasing attacks aim to rewrite the generated texts without changing the semantic meanings at the post-generation stage. Paraphrasing has been used for robustness evaluation and data augmentation in many other tasks, e.g., sentiment analysis, textual entailment (Iyyer et al., 2018), and machine translation (Merkhofer et al., 2022). Usually, an extra LLM is used as the paraphraser (Iyyer et al., 2018; Yang et al., 2022). Krishna et al. (2023) has reported the attack success of their paragraph-level paraphraser on some MGT detectors, but a comprehensive study across a wider range of paraphrasers and detectors is missing in the literature. In this section, we will introduce five attack types that cover paraphrasing attacks of different granularity, from word-level to paragraph-level.

6.3.1 Approaches

Synonyms Substitution is to replace some words with their synonyms to perturb the textual features. Inspired by the red teaming setting of Shi et al. (2023), we design a *model-free* method and a *model-based* method. For the model-free substitution, we replace the selected words with their

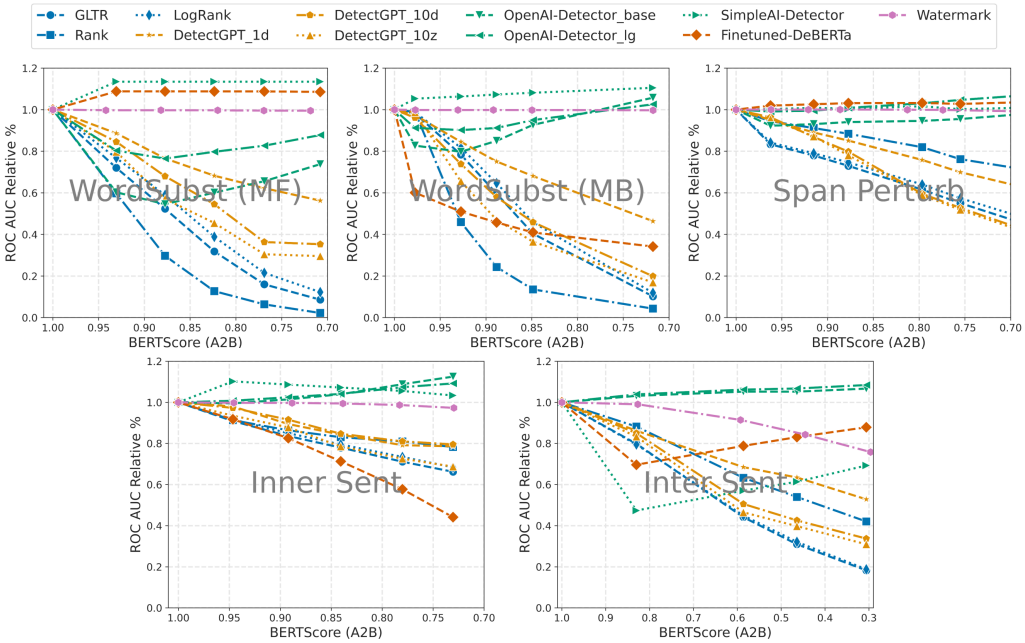

Figure 4: **Performance drop of the detectors under the paraphrasing attacks.** We use BERTScore (A2B) as the budget in the figure.'A2B' means we compute BERTScore between unattacked MGTs and attacked MGTs. ↓ Smaller BERTScore value means a larger budget on the attack.

synonyms retrieved from a static dictionary WordNet (Miller, 1994).[9] However, it does not consider the context of the substituted words. In the model-based method, we use T5-large (Raffel et al., 2020) to select the words to be substituted and prompt LlaMA (Touvron et al., 2023b) to get the synonyms given the context (detailed in Appendix F.3).

Span Perturbation is to rewrite word spans like phrases or clauses. Compared to synonym substitution, span perturbation is more flexible in that tokens can be reordered or replaced. Following the perturbation method of DetectGPT, we first randomly select spans for masking and then use T5-large to fill in.

Inner-Sentence Paraphrase is to paraphrase each sentence separately. We use Pegasus (Zhang et al., 2020) to process sentences of texts and join them back to the full texts. To control the budget, we can adjust the portion of sentences to be paraphrased.

Inter-Sentence Paraphrase uses Dipper (Krishna et al., 2023) to paraphrase the whole text at once, which can reorder, merge, and split multiple sentences. We control the lexical diversity and order diversity to change the budgets.

For budget, we measure the semantic difference between before- and after-attack with BERTScore.

[9]We avoid substituting the pronouns and prepositions to avoid grammatical problems. However, issues like verb tense still might happen.

6.3.2 Results and Analysis

The result is shown in Figure 4. **Interestingly, lower-level perturbations (i.e., word substitution) show greater attack success than higher-level perturbations (i.e., sentence-level paraphrases) at the same budget.** Metric-based detectors show weakness at all level perturbations, especially degrading to near 0.0 AUC ROC under word substitution attacks. Among metric-based detectors, DetectGPT shows slightly better robustness at word-level perturbation but then loses the lead at higher levels. For fine-tuned detectors, SimpleAI Detector remains robust under all attacks, while OpenAI Detectors and F.t.-DeBERTa fail at some attacks. A surprising result is that in some cases, fine-tuned detectors' performance first drops but then increases as the budget increases, e.g., OpenAI Detectors under word substitution and F.t.-DeBERTa under inter-sentence paraphrase. Finally, **for watermarking, inter-sentence paraphrasing is the only attack effective.**

See interpretation in Appendix D.2 and discussion on future defense in Appendix E.1.1.

6.4 Co-Generating Attacks

Co-generating attacks perturb the generated tokens at each recurrent step with some designed rules. Their mechanism shares similarities to the typo insertion attack (§6.2). But for co-generating, the perturbed text is cleaned to be grammatically cor-

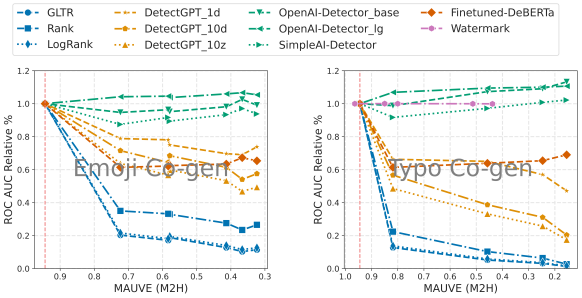

Figure 5: **Performance drop of the detectors under the co-generation attacks.** We use MAUVE (M2H) as the budget to evaluate the text quality in the figure.'M2H' means we compute MAUVE between HWTs and attacked MGTs. The vertically dotted red line is the score w.o. attack. ↓ Smaller MAUVE (M2H) value means a larger budget on the attack.

rect after generation. We will introduce two attack types: emoji and typo co-generation.

6.4.1 Approaches

Typo Co-Generation is to insert typos during generation. Different from the typo insertion attack (§6.2), we introduce typos immediately after the token is sampled (before the generation of the next token), following preset typo insertion rules, e.g., substitute all 'a's into 'z's. After the whole generation is finished, we reverse the inserted typos to clear the errors. Compared to the typo insertion attack, typo co-generation does not directly damage quality and human imperceptibility.

Emoji Co-Generation is developed in a similar fashion: We insert emojis at the end of generated sentences (before the generation of the next sentences) and remove them post-generation. The details are deferred to Appendix F.5.

6.4.2 Results and Analysis

Figure 5 shows the results. We observe that the metric-based detectors and F.t.-DeBERTa are not robust to the co-generation attacks, while OpenAI and SimpleAI Detectors show minor degradation. DetectGPT is more robust than other metric-based methods without perturbation, e.g., it converges at around 0.5 under typo co-generation while GLTR and (Log-)Rank converge near 0.1. For all detectors, the further increase in budgets for co-generation attacks does not cause proportional performance drops.

See interpretation in Appendix D.3 and discussion on future defense in Appendix E.1.3.

[10]For short, 'P.-Para' is the prompt paraphrasing attack, and 'CS Gen' is the character-substituted generation attack.

AUC%	Attack	*P.-Para*		*ICL*	*CS Gen*
	Dataset	*GPT-J*	*GPT-4*	*GPT-4*	*GPT-4*
	PPL unatt.	1.930	2.042	2.042	2.042
	MAUVE unatt.	0.944	0.483	0.483	0.483
Budget	PPL attacked	1.867	2.064	2.080	4.971
	MAUVE att.	0.963	0.348	0.680	0.056
Detect.	GLTR	105.3	111.3	96.83	**16.40**
	Rank	103.8	114.5	95.15	**13.47**
	LogRank	105.0	111.7	97.37	**16.58**
	DetectGPT-1d	99.64	109.4	98.96	**59.68**
	DetectGPT-10d	99.98	112.9	96.76	**31.44**
	DetectGPT-10z	99.94	112.9	97.15	**35.62**
	OpenAI Det.-Bs	115.9	135.8	96.71	**54.04**
	OpenAI Det.-Lg	110.4	128.1	99.79	**57.25**
	SimpleAI Det.	**25.63**	**33.20**	102.64	107.44
	F.t. DeBERTa	**43.70**	98.19	99.70	108.13
	Watermark*	99.98	- -	- -	- -

Table 5: **Performance drop of the detectors under the prompting attacks.** The perplexity (abbr. PPL) and MAUVE (M2H) are as the budgets for quality.[10]

6.5 Prompting Attacks

Most detectors are trained and tested on data based on fixed, well-designed prompts, e.g., question answering (Guo et al., 2023), continually writing (Zellers et al., 2019), etc. But in realistic scenarios, user prompts might be much more diverse, abnormal, and even noisy (Zamfirescu-Pereira et al., 2023). In this section, we introduce three attack types, namely prompt paraphrasing, in-context learning, and character-substituted generation. Table 5 shows the results. Due to lack of space, the detailed methods, results, analysis, and interpretation are delayed to Appendix B. Discussion on future defense is deferred to Appendix E.1.2.

7 Conclusion

Due to a lack of space, we defer future and related work to Appendix E and Appendix A.

This study evaluates the robustness of 8 MGT detectors against 12 realistic attacks, revealing striking vulnerabilities. Findings show that no detector consistently withstands all attacks, as some attack strategies severely compromise detection accuracy. Among various detectors, watermarking is the most robust, followed by model-based detectors. We also suggest combining metric- and model-based detectors for better resilience. Aiming at robust MGT detection, we call for awareness of vulnerability and the need for further methods.

Bolding indicates severe performance drop (drops larger than 50% are in red; between 25% and 50% are in yellow). Since GPT-4 is close-sourced, we can not test the watermark on it.

Limitations

We mainly show and discuss the results of representative generators, detectors, and attack methods in the main paper following the preset scope §2. Since our work is a general and reproducible evaluation pipeline, it is readily applicable to other generators or detectors.

We mainly focus on English in our work. Most attacks are able to be generalized to other languages, but the generation quality might suffer mainly depending on the generator's capability, especially in lower-resource languages. Also, the detection accuracy highly relies on the base model's capability in other languages. Some attacks could have slightly different designs for other languages, e.g., the homoglyph alteration attack could be more complex in logographic languages like Chinese, Japanese (Kanji), and Vietnamese (Chu Nôm), and it would be interesting to explore in future work.

Ethics Statement

The goal of this paper is not to provide a cookbook for malicious use of attacks to deceive MGT detectors. On the contrary, we want to draw attention to the potential vulnerabilities of current MGT detectors. Moreover, we call for future MGT detectors that are robust against the attacks we tested. For this target, we will open-source all the code and dataset for easy reproduction of our pipeline of robustness tests. We also propose and describe some defense patches for fixing these loopholes.

References

Serkan Ayvaz and Mohammed O. Shiha. 2017. The effects of emoji in sentiment analysis. *International Journal of Computer and Electrical Engineering*, 9:360–369.

Yukino Baba and Hisami Suzuki. 2012. How are spelling errors generated and corrected? a study of corrected and uncorrected spelling errors using keystroke logs. In *Annual Meeting of the Association for Computational Linguistics*.

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*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2022. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, 49:643–701.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2020. Language gans falling short. In *International Conference on Learning Representations*.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2022. Is GPT-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland. Association for Computational Linguistics.

Liam Dugan, Daphne Ippolito, Arun Kirubarajan, Sherry Shi, and Chris Callison-Burch. 2023. Real or fake text?: Investigating human ability to detect boundaries between human-written and machine-generated text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12763–12771.

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. *arXiv preprint arXiv:1903.11508*.

Shangbin Feng, Herun Wan, Ningnan Wang, Zhaoxuan Tan, Minnan Luo, and Yulia Tsvetkov. 2024. What does the bot say? opportunities and risks of large language models in social media bot detection. *arXiv preprint arXiv:2402.00371*.

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

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

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

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Tianxing He, Jingyu Zhang, Tianle Wang, Sachin Kumar, Kyunghyun Cho, James Glass, and Yulia Tsvetkov. 2023. On the blind spots of model-based evaluation metrics for text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12067–12097, Toronto, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2023. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*.

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

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew A. Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84:414–420.

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.

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

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24:377–439.

Sachin Kumar, Vidhisha Balachandran, Lucille Njoo, Antonios Anastasopoulos, and Yulia Tsvetkov. 2022. Language generation models can cause harm: So what can we do about it? an actionable survey. *arXiv preprint arXiv:2210.07700*.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2953–2965, Dublin, Ireland. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. 2024. Does {DetectGPT} fully utilize perturbation? selective perturbation on model-based contrastive learning detector would be better.

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*.

Chengzhi Mao, Carl Vondrick, Hao Wang, and Junfeng Yang. 2024. Raidar: generative ai detection via rewriting. *arXiv preprint arXiv:2401.12970*.

Elizabeth M. Merkhofer, John Henderson, A. N. Gertner, Michael Doyle, and Lily Wong. 2022. Practical attacks on machine translation using paraphrase. In *Conference of the Association for Machine Translation in the Americas*.

George A. Miller. 1994. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

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

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*.

Moin Nadeem, Tianxing He, Kyunghyun Cho, and James Glass. 2020. A systematic characterization of sampling algorithms for open-ended language generation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 334–346, Suzhou, China. Association for Computational Linguistics.

OpenAI. 2022a. Chatgpt. Website.

OpenAI. 2022b. Text-davinci-003. Website.

OpenAI. 2023. Gpt-4 technical report.

Artidoro Pagnoni, Martin Graciarena, and Yulia Tsvetkov. 2022. Threat scenarios and best practices to detect neural fake news. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1233–1249, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Mička Pavel. 2000. Letter frequency (english). *The Mathematical Association of America*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Xiao Pu, Jingyu Zhang, Xiaochuang Han, Yulia Tsvetkov, and Tianxing He. 2023. On the zero-shot generalization of machine-generated text detectors. *arXiv preprint arXiv:2310.05165*.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. 2023. Red teaming language model detectors with language models. *arXiv preprint arXiv:2305.19713*.

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

Richard J. Spillman, Mark Janssen, Bob Nelson, and Martin Kepner. 1993. Use of a genetic algorithm in the crypt-analysis of simple substitution ciphers. *Cryptologia*.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. *arXiv preprint arXiv:2306.05540*.

Mirac Suzgun, Stuart M. Shieber, and Dan Jurafsky. 2023. string2string: A modern python library for string-to-string algorithms.

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

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

Yi-Ting Tsai, Min-Chu Yang, and Han-Yu Chen. 2019. Adversarial attack on sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 233–240, Florence, Italy. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, et al. 2023. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. *arXiv preprint arXiv:2305.14902*.

Jonatas Wehrmann, Mauricio A. Lopes, Douglas M. Souza, and Rodrigo C. Barros. 2019. Language-agnostic visual-semantic embeddings. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5803–5812.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Kexin Yang, Dayiheng Liu, Wenqiang Lei, Baosong Yang, Haibo Zhang, Xue Zhao, Wenqing Yao, and Boxing Chen. 2022. GCPG: A general framework for controllable paraphrase generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4035–4047, Dublin, Ireland. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qiang Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*.

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

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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

Yi-Fan Zhang, Zhang Zhang, Liang Wang, and Rong Jin. 2023. Assaying on the robustness of zero-shot machine-generated text detectors. *arXiv preprint arXiv:2312.12918*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023a. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023b. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*.

Unwatermarked		*Watermarked*	
PPL	*MAUVE(M2H)*	*PPL*	*MAUVE(M2H)*
1.930 ± 0.386	0.9444	2.119 ± 0.524	0.9639

Table 6: The unattacked value of average Perplexity and MAUVE (M2H) as the base point. Notably, for the watermark-based detector, the reference texts for budget computation are watermarked MGTs instead of the original unwatermarked MGTs.

A Related Work

To the best of our knowledge, there is no existing thorough study on the robustness of machine-generated text detection under various attacks. The most related studies are:

Study on Adversarial Attack to MGT Detection. Adversarial attack (Goodfellow et al., 2014), which exposes optimized regions of the input space where the model under-performs, first introduced to text data by Li et al. (2018), is powerful to reveal robustness in text classification (Jin et al., 2020). Shi et al. (2023) first use adversarial attack on MGT detectors, including OpenAI-Detector, DetectGPT, and watermarking. They cover adversarial word substitution and adversarial prompting, both of which deceive three detectors. Furthermore, RADAR (Hu et al., 2023) attempts to improve the robustness of model-based detectors by adversarial learning on paraphrasing. We also take inspiration from recent work on the blind spots of NLG metrics (He et al., 2023).

However, under realistic scenarios, attackers do not have detailed knowledge of which detector is being used (§2). Our work focuses on non-adversarial attacks, which are less costly and under-explored.

Study on Generalization of MGT Detection. Generalization capability is an important aspect of robustness in MGT detection. For model-based detectors, Solaiman et al. (2019) evaluate their OpenAI Detector on generalize through different model sizes, sampling strategies, and input text length. Pu et al. (2023) study the generalization ability when training and testing on data from different generators. Pagnoni et al. (2022) analyze the generalization on sequence length, decoding strategy, dataset domain, and generator size. Wang et al. (2023) introduce a multi-generator, multi-domain, and multi-lingual corpus to train more generalizable detectors. For metric-based detectors, Mireshghallah et al. (2023) explore the generalization between different base models and dataset generators on a perturbation-based metric-based detector. In comparison, our research focus is not on the generalization problem but on the robustness against realistic and malicious attacks.

Robustness of MGT Detection. Some existing works of MGT detectors evaluate their robustness under some specific attacks. Liu et al. (2022) evaluate the robustness of their model-based detector CoCo under token editing. Krishna et al. (2023) stress test detectors on paragraph-level paraphrase, and further purpose a retrieval-based method to increase robustness. Hu et al. (2023) focus on paraphrastic robust model-based detectors by adopting adversarial learning. Zhang et al. (2023) purpose that topic shifting drops the metric-based detectors' performance. In the watermark domain, Kirchenbauer et al. (2023a) propose a list of initial attack ideas, including editing, paraphrasing, and generation strategy. But, they only experiment on the span perturbation attack for their watermark method. Further, Kirchenbauer et al. (2023b) study the watermark robustness after LLM paraphrase, manual paraphrase, and mix into a longer document. Kuditipudi et al. (2023) purpose a distortion-free watermark that is robust against perturbation. Zhao et al. (2023b) enhance the robustness of the watermarking scheme against editing and paraphrasing attacks by employing a fixed group design. And Hou et al. (2023) propose a semantic watermark at the sentence level for paraphrastic robustness. To summarize, a thorough and comparative study on the robustness covering a wide range of detectors and attacks is lacking in the literature, which motivates our work.

B Prompting Attack

B.1 Approaches

Prompt Paraphrasing. Instead of paraphrasing whole texts post-generation (§6.3), paraphrasing the prompt prior to generation is much cheaper and low-impact on the output quality. We use Pegasus paraphraser to rewrite the prompts.[11]

In-Context Learning (Dong et al., 2022) improves generation quality by giving only a few examples in the form of demonstration. To generate more human-like texts to deceive detectors, we give the

[11] Since the prompts are usually very short, it is hard for us to control the budget while paraphrasing. Hence, we report attacked performance under a single budget at Table 5.

Prompt:	Continue 20 words with all 'a's substituted with 'z's and all 'z's substituted with 'a's: As the sun dipped below the horizon, casting
GPT-4:	Zs the sun dipped below the horiaon, czsting shzdows zcross the lzndsczpe, z gentle breeae whispered through the trees, czrrying with it the sweet zromz of spring flowers ...
Cleaned Output:	As the sun dipped below the horizon, casting shadows across the landscape, a gentle breeze whispered through the trees, carrying with it the sweet aroma of spring flowers ...

Table 7: A character-substituted generation example.

generator a related HWT as a positive example and a vanilla MGT as a negative example. We follow the prompt design of Super-NaturalInstructions (Wang et al., 2022).[12]

Character-Substituted Generation is inspired by substitution cipher (Spillman et al., 1993) in the domain of cryptography. We found that recent LLM, e.g., GPT-4, are able to follow some easy substitution rule, e.g., `replace 'a's with 'z's`, during generation. Afterward, we substitute the mapping back to clean the output. Table 7 shows an example. Notably, we only find GPT-4 maintains acceptable generation quality under this attack, so we solely show its result.

B.2 Results and Analysis

Table 5 shows the results. The metric-based detectors severely suffer the character-substituted generation attack but are robust to prompt paraphrasing and in-context learning. On the other hand, the fine-tuned detectors show some drop in prompt paraphrasing but have great robustness under character-substituted generation.[13]

Interpretation. The character-substituted generation attack is a more localized perturbation compared with prompt paraphrasing and in-context learning, which is on the general level. So, similar to the paraphrasing attacks, metric-based detectors show a larger vulnerability to localized perturbation since it directly increases the next-token probabilities, which is also shown as the high perplexity after the attack. However, fine-tuned detectors focus more on long-term patterns, which may not impacted by a few substitutions. But, prompt paraphrasing is a form of attack that shifts the prompt pattern, which can degrade fine-tuned detectors severely, especially those ones that are not generalizable.

Discussion on future defense is deferred to Appendix E.1.2.

[12] It is also hard to adjust the budget for this attack. One potential way is to change the demonstration number, but it shows no clear correlation to the budgets and also might exceed the generator's maximum length of the input sequence.

[13] Note that the budget of character-substituted generation is larger than other attacks. As a prompting method, it is hard to control it, so a milder character-substitution method with an adjustable budget is by controlled generation (§6.4).

C Experiment Settings

The experiments are done on 8 Tesla V100 and 4 Tesla A100 GPUs, taking up a total of around 500 GPU hours.

C.1 Dataset and Generators

We build the dataset based on Pu et al. (2023). The HWTs are from the News domain of the dataset, and the MGTs are generated with different temperatures for each generator we selected. Table 8 shows the sample number of each split in our dataset.

Split	Train	Eval	Test
Sample Num.	8,000	1,000	1,000

Table 8: The sample number of each split of the dataset.

For sampling, we use a combination of nucleus sampling (Welleck et al., 2019) with top-p = 0.96 and a tuned temperature parameter (Caccia et al., 2020; Nadeem et al., 2020). While smaller temperature gives higher quality, it will also cause repetition, especially for less capable LMs. So, we tune the temperature based on the criteria of preventing repetition, which is < 0.2 in terms of 4-gram duplication under metric seq-rep-4 in Welleck et al. (2019). Table 9 shows the our temperature settings.

`Generator`	GPT-2 XL	GPT-J	LlaMA	LlaMA-2	DaVinci-003	GPT-4
Temp.	1.5	1.5	1.0	1.5	0.7	0.7

Table 9: The temperature we set for each generator to follow the criteria of avoiding severe repetitions.

C.2 Detectors

C.2.1 Detailed Introduction

Metric-Based Detector relies on the inferenced log-probability from the generator LLM, and adopts a threshold for classification.

GLTR (Gehrmann et al., 2019; Solaiman et al., 2019) using the average of the next-token proba-

bility to determine whether an input text is MGT. Texts with high average probability are classified as MGTs.

Rank and *LogRank* (Solaiman et al., 2019; Mitchell et al., 2023) using the averaged rank and log-rank of next-token probability for detection respectively.

DetectGPT (Mitchell et al., 2023) stands as the pioneering work of using perturbation as a comparison to original texts to enhance metric-based detection. Perturbation here refers to rewriting or substituting spans of tokens using a mask-filling LM (i.e., T5-small (Raffel et al., 2020)). It poses that perturbed MGTs tend to have lower log probabilities compared to the original samples under the base LM, while perturbed HWTs may be at about a similar level to the origin. Bao et al. (2023); Su et al. (2023); Liu et al. (2024); Mao et al. (2024) further follow up DetectGPT.

We apply the white-box setting to the metric-based detectors, where full knowledge (e.g., which LLM generated the texts) and access (including the parameters of the generator LLM) are given to the detectors. The reason is that those detectors require the generator LLM as the base model to compute the metrics.

Fine-Tuned Detector is trained on a pretrained language model (PLM) in a supervised method with a classification loss.

OpenAI Detector (Solaiman et al., 2019) is a model to detect GPT-2 generation by fine-tuning a RoBERTa (Liu et al., 2019) model. We evaluate both the base size (125M) and the large size (355M) model.

SimpleAI Detector (Guo et al., 2023) is a detector mainly for distinguishing ChatGPT, using the HC3 QA dataset (Guo et al., 2023) to fine-tune a RoBERTa model.

Fine-tuned DeBERTa is the model we fine-tuned on our generation data, representing an in-domain setting. We use DeBERTa-v3-base (He et al., 2021) as the base model.[14]

Compared with OpenAI and SimpleAI Detectors as off-the-shelf models, our fine-tuned DeBERTa is relatively in-domain since it is solely fine-tuned on the dataset from the same generator and within the same topic domain as the test set. All the fine-tuned detectors are under the black-box setting, which means they have no knowledge or access to the generator LLM but only the generated dataset.

Watermark-Based Detector adds algorithmically detectable signatures into texts during generation. Kirchenbauer et al. (2023a) is a representative watermarking approach, which adds a token-level bias in the decoding stage (represented as *Watermark* afterward). This work is followed up by Zhao et al. (2023b); Christ et al. (2023); Kuditipudi et al. (2023); Hou et al. (2023). All watermark-based detectors are under the white box setting, where they have all the knowledge and access to the generator LLM.

C.2.2 Detailed Hyperparameters

For all model-based detectors, we use the original generator of the test set as the base model to compute the next-token probability and perplexity.

For *DetectGPT*, we follow the recommendation hyperparameter setting. The perturbation word ratio is 15% on 2-spam, the perturbation model is T5-3B (Raffel et al., 2020), and the sample number of perturbation is 1 or 10 (indicated in the name of the legend). In the legend, mode 'd' represents the direct use of the absolute likelihood drop while mode 'z' adds an additional normalization. The mask-filling in perturbation is with temperature 1 without any sampling strategy (e.g., top-p and top-k).

For all fine-tuned detectors, we directly use the logits as the output probability. When fine-tuning *F.t. DeBERTa*, we set batch size as 4, learning rate as 1e-5, weight decay as 0, adam epsilon as 1e-8, and epoch number as 10.

For the watermark, we follow the setting in Kirchenbauer et al. (2023a), setting gamma as 0.25, seeding scheme as selfhash, and z-score threshold as 4.0.

C.2.3 Under Closed-Source Dataset

For GPT-4 datasets, as we do not have the white-box generator model, we select an alternative LM as the base model. According to the conclusion from Mireshghallah et al. (2023), GPT-2 Small (Radford et al., 2019) is the best-performed base model when generalized to GPT-4. Our experiment compares GPT-2 (Small, Medium, Large, XL), OPT (125M, 350M, 1.3B, 2.7B) (Zhang et al., 2022), GPT-Neo (125M, 1.3B, 2.7B) (Black et al., 2021), and GPT-J (6B), and the results align that GPT-2 Small is the best. Hence, our GPT-4 dataset

[14] We have also tried other base models, e.g., BERT (Devlin et al., 2018), RoBERTa, ELECTRA (Clark et al., 2020), etc. The selection of base model does not impact the overall trend of the findings, and the gap on the absolute detection accuracy is within 2%.

results are all under GPT-2 Small as the base model. Table 11 shows the unattacked performance. Table 10 shows the Perplexity and MAUVE (M2H) as budget of unattacked GPT-4 dataset.

Unwatermarked	
PPL	*MAUVE(M2H)*
2.042 ± 0.250	0.4831

Table 10: The unattacked value of average Perplexity and MAUVE (M2H) of GPT-4 dataset as the base point budget.

Detector	*AUC*	*TF=5*	*TF=10*	*TF=20*	*ACC*
GLTR	62.41	2.20	7.20	22.00	60.40
Rank	62.15	11.40	21.20	36.00	59.80
LogRank	65.96	5.00	17.00	32.60	62.00
Entropy	66.40	12.40	23.00	35.80	61.80
DetectGPT-1d	51.22	3.60	6.60	15.20	50.40
DetectGPT-10d	55.61	2.00	4.40	16.20	55.60
DetectGPT-10z	59.53	5.80	11.00	22.20	58.40
OpenAI Det.-Bs	55.72	13.20	20.20	28.60	52.60
OpenAI Det.-Lg	57.70	6.00	12.20	24.60	56.00
SimpleAI Det.	86.81	81.00	82.20	85.40	84.40
F.t. DeBERTa	100.0	99.80	99.80	99.80	99.80

Table 11: **The performance of detectors in the unattacked scenario for the GPT-4 dataset.** For short, 'AUC' is ROC AUC, 'TF=5' is TPR@FPR=5%, 'ACC' is Accuracy, 'Det.' is Detector, and 'F.t.' is Fine-tuned.

D Interpretation

D.1 Editing Attacks (§6.2)

For the fine-tuned detectors, OpenAI Detectors perform worse in most cases, while SimpleAI Detector and F.t.-DeBERTa show great robustness. We surmise the reason is that OpenAI Detectors is trained on the GPT-2 corpus, which is outdated compared to the ChatGPT corpus for SimpleAI Detector and the GPT-J corpus for F.t.-DeBERTa. Under such an out-of-distribution (OOD) situation, the model shows less robustness.

The watermarked detector adds a signature at each token, and our editing attacks only change a minimal portion of them. Hence, they show substantial robustness, maintaining high AUC ROC.

D.2 Paraphrasing Attacks (§6.3)

For metric-based detectors, localized disturbances from lower-level perturbations cause more decreases in next-token probability than high-level perturbations. While for high-level perturbations, the decrease is spread out in wider spans, thus minor the overall impact. For fine-tuned detectors, Liu et al. (2022) pose that they concentrate more on long-form patterns (e.g., commonly used phrases or sentence structures) from LLM to detect. Hence, localized disturbances of low-level perturbation directly interrupt the long-form patterns, while high-level paraphrasing is milder as it rewrites such patterns but still keeps some of the machine signatures. Moreover, we surmise that paraphrasing attacks are not making MGTs more human-like but only mixing the machine signatures. So, sometimes, the detectors' performance falls then rises as the budgets increase, during which the dominant machine signatures switch from the original generator's to the paraphraser's.

D.3 Co-Generating Attacks (§6.4)

The insertion of emojis and typos during recurrent next-token generation is a disruption for the sampling of LLMs, shifting the generation away from the generator's original distribution. Moreover, removing the emojis and recovering the typos post-generation disrupt the conditional probability again for metric-based detectors. For fine-tuned models, we surmise that when doing in-domain detection (F.t.-DeBERTa), the detector might focus more on localized features. Otherwise, out-of-domain models here (OpenAI and SimpleAI Detectors) focus on long-term patterns. Thus, the in-domain model is less robust to the attacks.

We have also attempted emoji co-generation for watermarking, and it also demonstrates very strong robustness, similar to the typo case. Interestingly, inserting more emojis did not affect the budget (MAUVE score) for watermarked generation. Therefore, we choose not to plot this result in Table 5 to avoid confusion.

E Future Work

E.1 Future Work on Defenses

E.1.1 Paraphrasing Attacks (§6.3)

For metric-based detectors, a straightforward way is to choose a base model that is related to common paraphrasers' base models, e.g., T5, ProphetNet (Qi et al., 2020), or fine-tune the base model on some paraphrased corpus. Similarly, data augmentation on paraphrasing and adversarial learning could be useful for training fine-tuned detectors (Hu et al., 2023). Moreover, Krishna et al. (2023)

purpose that retrieval on an MGT database can be the defense, if it is possible to collect enough in-domain MGT entries. For watermarking, semantic-level watermarking (Hou et al., 2023) (as opposed to token-level) has been proposed for paraphrastic robustness.

E.1.2 Prompting Attacks (§6.5)

To patch the weakness of fine-tuned detectors under prompt paraphrasing, an efficient way is to fine-tune the classifier on multi-generator, multi-domain datasets, e.g., M4 by Wang et al. (2023). Otherwise, using an ensembling system (Pagnoni et al., 2022) containing both metric-based and fine-tuned detectors could ease the problem. However, we surmise there is no direct way to patch the character-substituted generation because it mimics the suboptimal generation strategy of humans at the root. Yet current LLMs are not capable of always following the character-substitution prompts with high text quality, which could cause unnatural expressions and extra typos. A way to fix the loophole could be censoring the prompts and generated texts if they have weird expressions (Dou et al., 2022; Chiang and Lee, 2023). Additionally, training detectors on the MGT corpus from unnatural instructions (Honovich et al., 2022) could also be considered.

E.1.3 Co-Generating Attacks (§6.4)

To the best of our knowledge, there are no related existing works. We feel it is hard for metric-based detectors to overcome the defects. Under this scenario, fine-tuned detectors could be the better choice. One potential way to enhance fine-tuning is to adopt some data augmentation, like random masking on short-term spans. Also, we surmise a combination of metric-based detectors and model-based detectors is useful to bypass each other's stumbling blocks better when attacked. The ensembling could also ease the impact of other attacks. Fortunately, the co-generation attacks are still not widely available now since they need to be on the white-box models.

E.2 Future Work on Attacks

Below, we briefly discuss other types of attacks related to generalization, which are not covered in this work.

Sampling Attacks. Diverse sampling strategies (Holtzman et al., 2019) can be adopted when generating MGTs both by setting different hyper-parameters. Pagnoni et al. (2022) show that detection performance generally decreases when a fine-tuned detector is evaluated on a sampling strategy it was not trained on.

AUC%	Typo Type	*Mixed*	*Insert*	*Delete*	*Subst.*	*Trans.*
Budget	Edit Distance	17.68	18.05	18.04	16.76	17.87
Detect.	GLTR	2.14	2.96	6.96	3.17	5.76
	Rank	6.81	7.25	13.70	6.67	12.18
	LogRank	2.56	3.65	9.74	3.72	7.67
	DetectGPT-1d	44.66	44.57	53.38	42.59	58.28
	DetectGPT-10d	17.99	15.98	32.62	18.01	25.18
	DetectGPT-10z	15.02	14.54	26.24	15.95	20.75
	OpenAI Det.-Bs	27.62	27.37	24.00	26.32	25.57
	OpenAI Det.-Lg	34.76	29.56	35.11	32.68	33.58
	SimpleAI Det.	111.6	111.1	112.1	111.0	111.2
	F.t. DeBERTa	108.4	96.80	97.20	96.83	97.48

Table 12: Detectors' performance drops in terms of relative AUC ROC % of 4 typo types, namely insertion, deletion, substitution, and transposition.

Fine-Tuning Attacks. In some scenarios, users might fine-tune the generator LLM on their specific domain. Since the detectors have no knowledge and access to the customized generator, their performance might decrease.

Human-Involved Attacks is to manually polish or replenish MGTs to be more human-like and improve their quality, which could deceive the MGT detector. Kirchenbauer et al. (2023b) purpose manual paraphrasing and mixing HWTs into MGTs as an attack to watermarks. And Christ et al. (2023) describe a manual prefix-specificity scheme to lead to a more human-like generation. Therefore, a major limitation of the current detector technique is the inability to classify human-LLM-collaborated texts into binary classes. Future MGT detectors that are able to measure the portion of LLM involvement in text writing are worth considering as an answer to this attack genre.

F Attack Details

In this section, we report the details that are not included in the main paper due to lack of space, including methodologies and settings.

F.1 Typo Insertion

Table 12 shows the performance drop of four separate typo types, i.e., insertion, deletion, substitution, and transposition. All of them share similar observations on degradation trends and are close to the mixed typo type. Therefore, for the figure in the main text, we show the result of mixed for brevity.

F.2 Homoglyph Alteration

We consider `\n` - newline, `\r` - carriage return, `\v` - vertical tab, `\u200B` - zero-width whitespace, and `\u000B` - line tabulation as representatives and they shared similar results. Specifically, zero-width whitespace can be inserted between any tokens, while we only add shift-related characters at the end of sentences.

F.3 Synonym Substitution

Table 13 shows the prompt design for LlaMA to do the model-based synonym generation with the context. After the generation, we have an additional step to ask LlaMA double check and correct the grammar of the substituted sentences.

```
${sentence}\n Synonyms of the word "${word}"
in the above sentence are:\n a)
```

Table 13: Prompt for LlaMA to generate synonyms based on the context for substitution attack.

F.4 Typo Co-Generation

The results reported in the main text using the typo substitution rule switching 'c's and 'k's. We have also tried other rules, e.g., 'a's and 'z's. The different rules cause different budgets depending on the character appearance frequency in the texts. We select a rule that has a comparable budget interval to other attacks, but our system also supports other rules.

F.5 Emoji Co-Generation

Emojis are widely used in web texts, especially social media (Ayvaz and Shiha, 2017). However, emojis are usually excluded from the training corpus of fine-tuned detectors and are situated at the long tail of distribution for metric-based detectors. Thus, they have a similar effect as the insertion of typos (§6.2). We insert a random emoji from Gemoji[15] when LLM finishes a sentence and let the LLM generate the next sentence recurrently. We control the budgets by tuning the probability of inserting an emoji after a sentence. We clean the output texts after generation by removing all emojis to hide the trace of the attack. Note that the distribution shift caused by emoji during sampling will still embodied in the text and deceive the detectors.

[15] A package of emoji collections: https://github.com/wooorm/gemoji.

G Additional Results

G.1 Across Budgets

The design of the budget considers the alignment of different metrics's indications, especially for the ones on the same aspects.

Figure 6 to Figure 10 and Figure 11 to Figure 15 show the performance drop in terms of BERTScore, BARTScore, Cosine Similarity, Jaro Similarity, and Edit Distance for paraphrasing attacks. Figure 16 to Figure 17 show the editing attacks, and Figure 18 to Figure 21 show the co-generating attacks. The line charts illustrate a similar trend for performance drop of MGT detectors under attacks, which cross-validate our results and conclusion. Also, they support the reasonability of the design of our budget.

G.2 Across Metrics

Figure 22 to Figure 27 shows the performance drop of the detectors in terms of different metrics, including ROC AUC, PR AUC, accuracy (ACC), TPR@FPR=20%, =10%, and =5%. The similar drop trends show the correlation between all metrics involved in our study. Here, we show editing attacks as examples and omit others for brevity.

G.3 Across Generators

In this section, we report the results of the main test on LlaMA-2 (Touvron et al., 2023a) as the generator. As we have mentioned, due to larger LLMs not having good detection capability for metric-based detectors (Mireshghallah et al., 2023), the trend results might be noisy and unclear compared with the GPT-J main results in §6. However, the results and conclusion align well across generations at a high level. Table 14 and Figure 28 - Figure 28 show the results.

Figure 6: Performance drop under the paraphrasing attacks with **BERTScore** (A2B) as budget (x-axis). (Row 1)

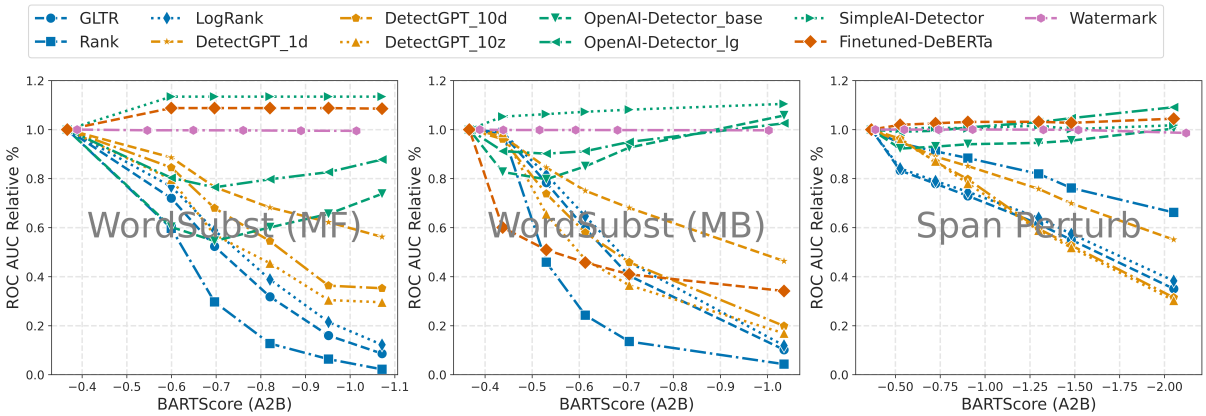

Figure 7: Performance drop under the paraphrasing attacks with **BARTScore** (A2B) as budget (x-axis). (Row 1)

Detector	*AUC*	*TF=5*	*TF=10*	*TF=20*	*ACC*
GLTR	84.09	29.60	52.20	72.00	76.40
Rank	67.15	17.80	29.20	42.80	64.60
LogRank	87.25	40.20	62.60	78.60	79.20
Entropy	46.96	6.20	10.00	21.80	47.80
DetectGPT-1d	57.83	5.00	12.60	26.00	54.20
DetectGPT-10d	66.26	15.40	22.20	38.40	61.00
DetectGPT-10z	72.91	16.20	33.00	52.00	66.40
OpenAI Det.-Bs	74.40	30.20	40.60	53.40	68.20
OpenAI Det.-Lg	79.62	31.40	41.00	62.60	72.60
SimpleAI Det.	88.26	82.00	83.40	85.80	84.80

Table 14: **The performance of detectors in the unattacked scenario for the LlaMA-2 dataset.** For short, 'AUC' is ROC AUC, 'TF=5' is TPR@FPR=5%, 'ACC' is Accuracy, 'Det.' is Detector, and 'F.t.' is Fine-tuned.

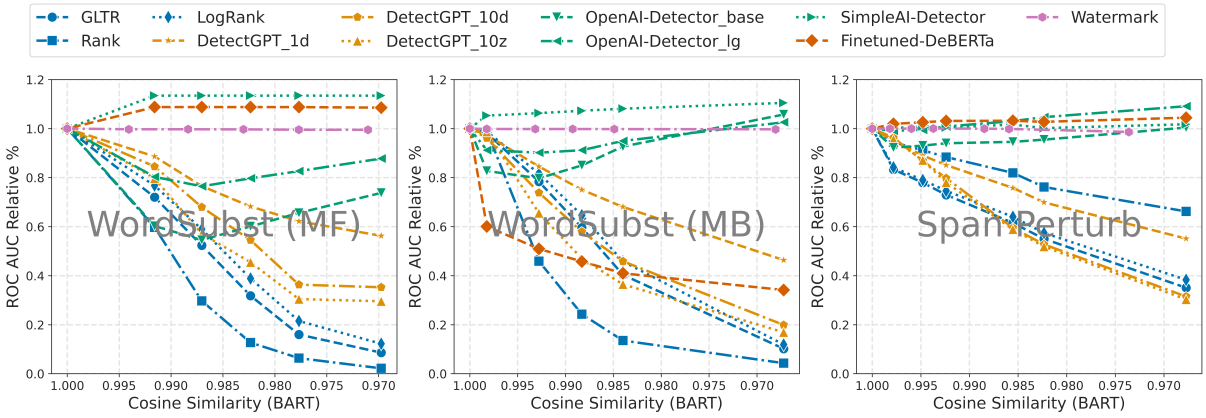

Figure 8: Performance drop under the paraphrasing attacks with **Cosine Similarity** as budget (x-axis). (Row 1)

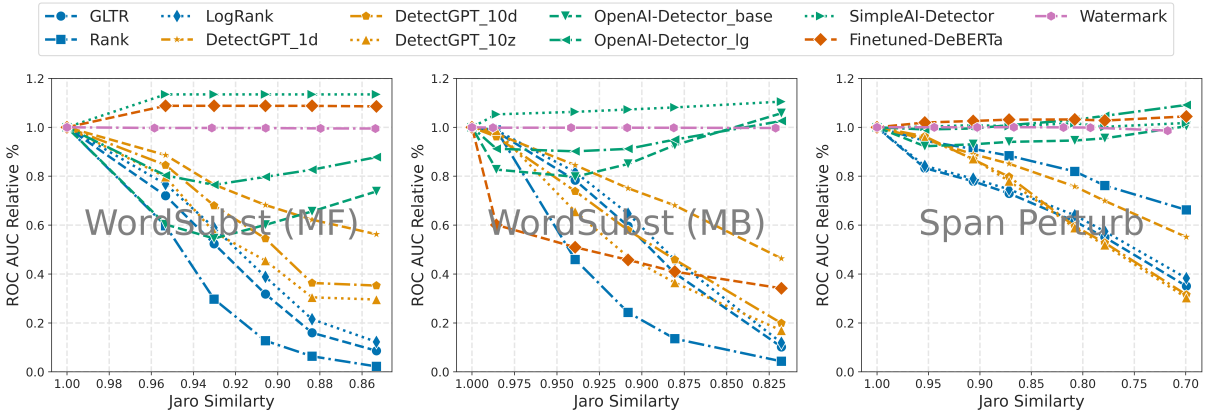

Figure 9: Performance drop under the paraphrasing attacks with **Jaro Similarty** as budget (x-axis). (Row 1)

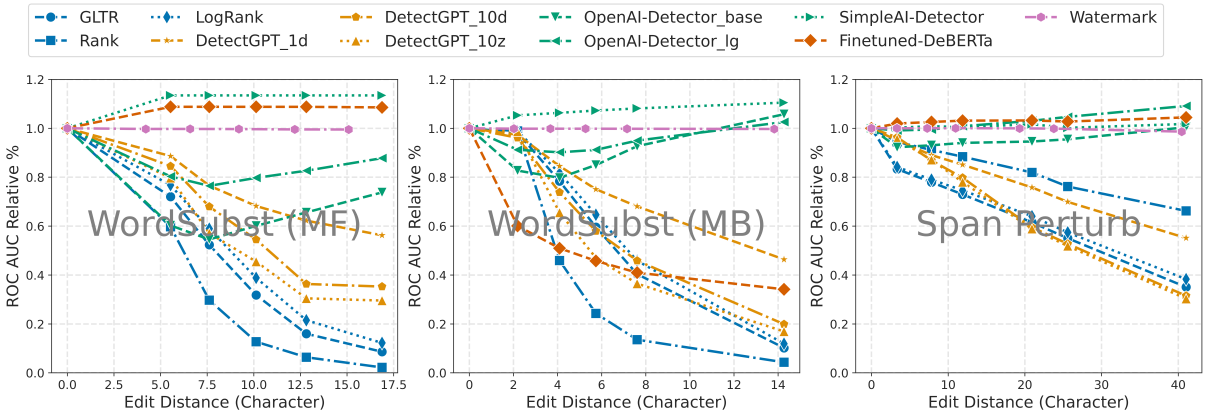

Figure 10: Performance drop under the paraphrasing attacks with **Edit Distance** as budget (x-axis). (Row 1)

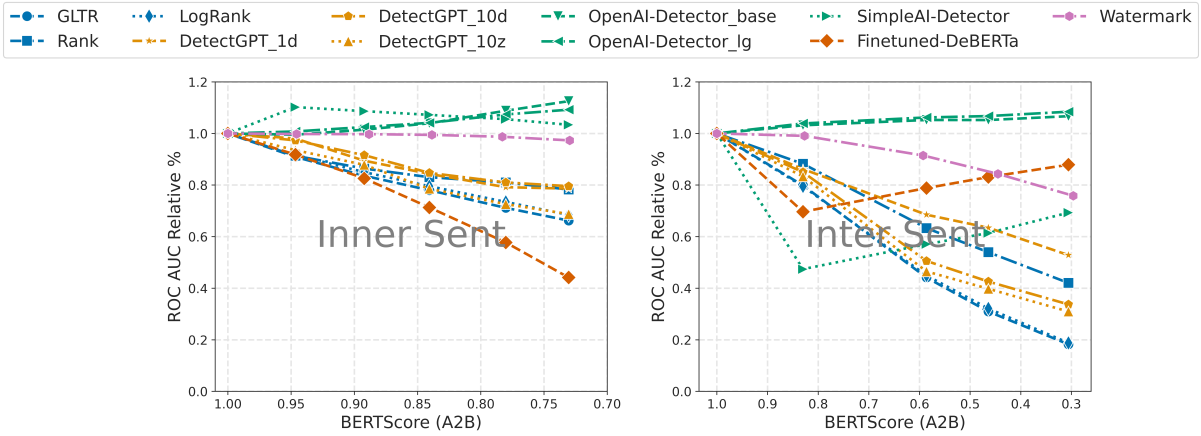

Figure 11: Performance drop under the paraphrasing attacks with **BERTScore** (A2B) as budget (x-axis). (Row 2)

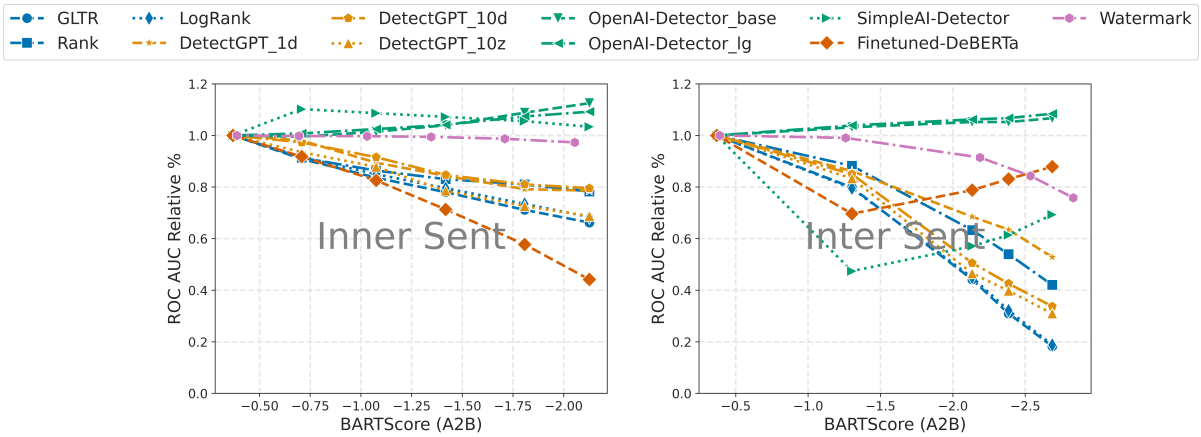

Figure 12: Performance drop under the paraphrasing attacks with **BARTScore** (A2B) as budget (x-axis). (Row 2)

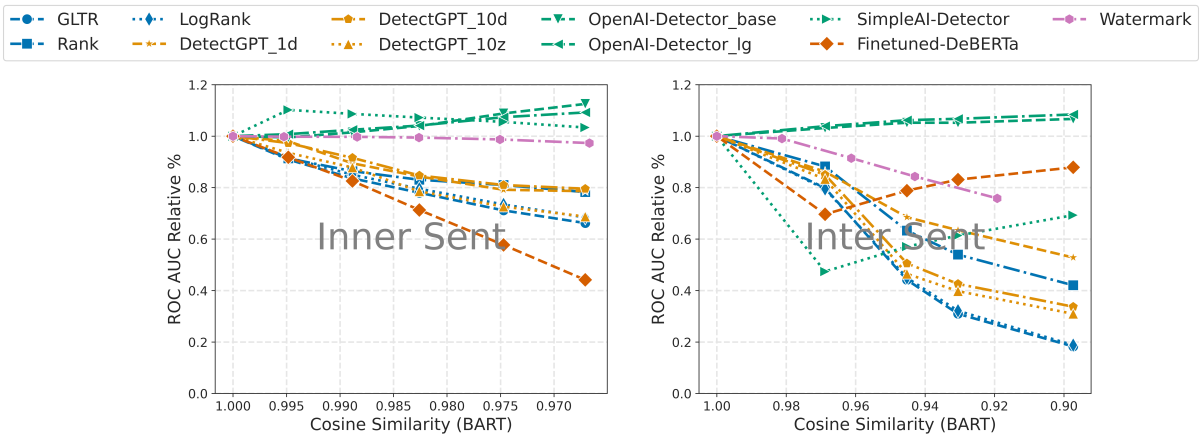

Figure 13: Performance drop under the paraphrasing attacks with **Cosine Similarity** as budget (x-axis). (Row 2)

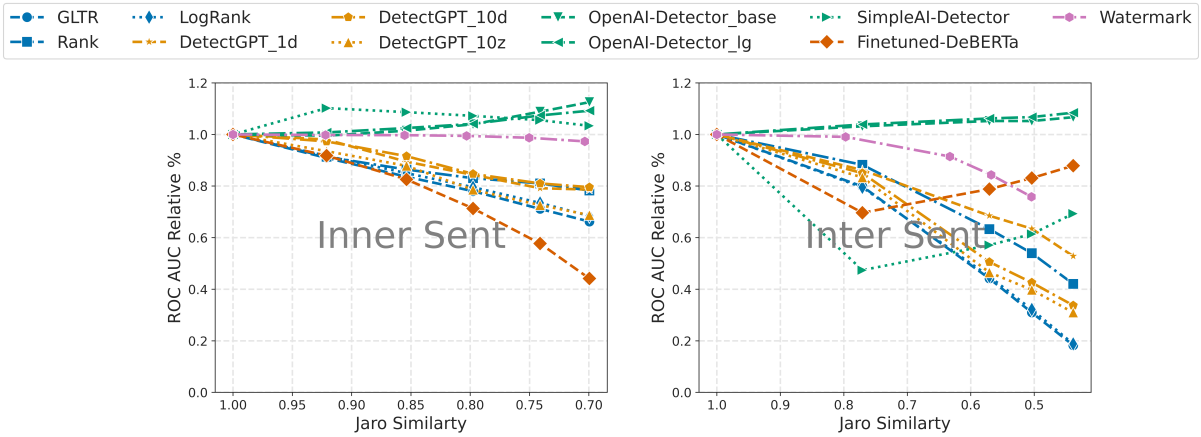

Figure 14: Performance drop under the paraphrasing attacks with **Jaro Similarty** as budget (x-axis). (Row 2)

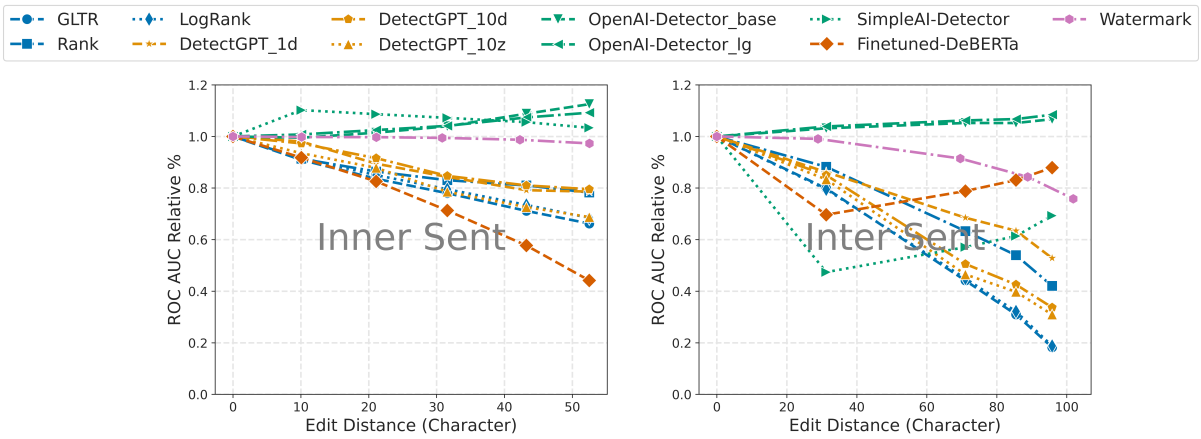

Figure 15: Performance drop under the paraphrasing attacks with **Edit Distance** as budget (x-axis). (Row 2)

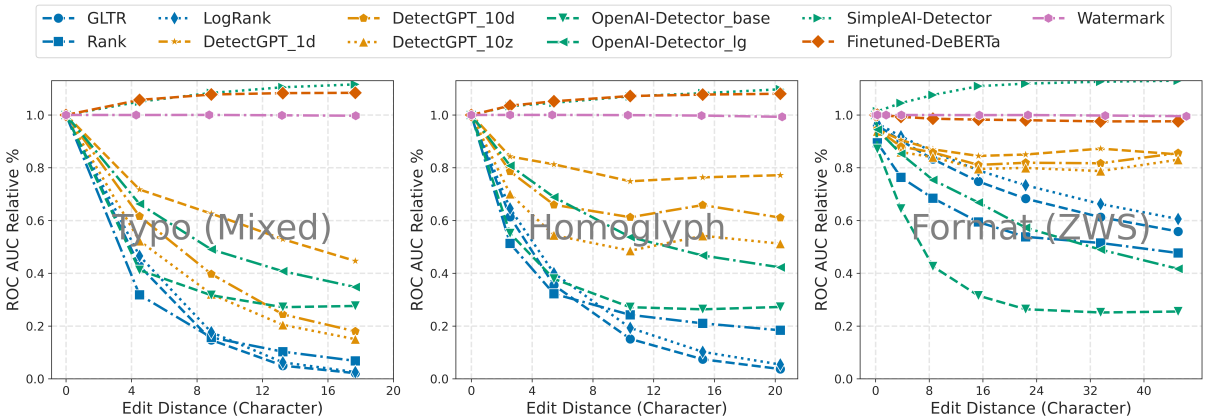

Figure 16: Performance drop under the editing attacks with **Edit Distance** as budget (x-axis).

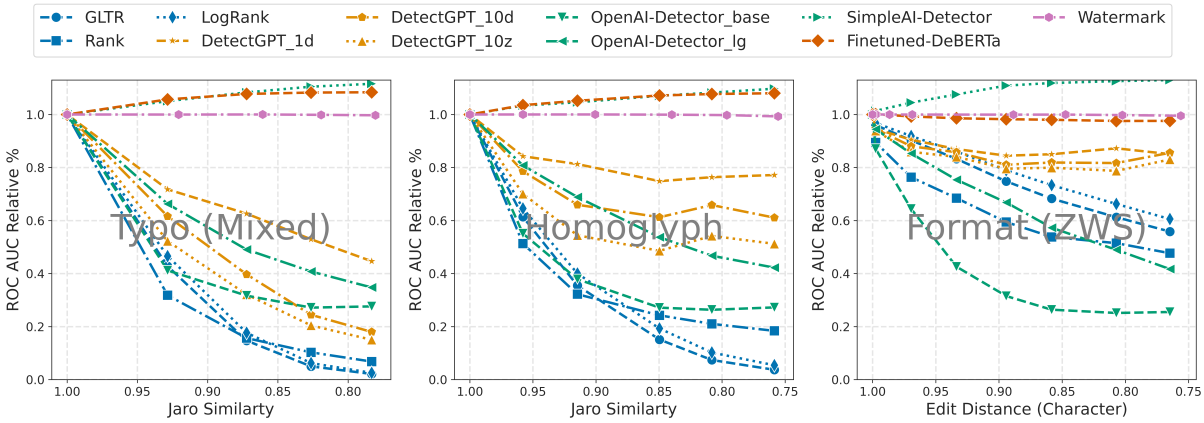

Figure 17: Performance drop under the editing attacks with **Jaro Similarty** as budget (x-axis).

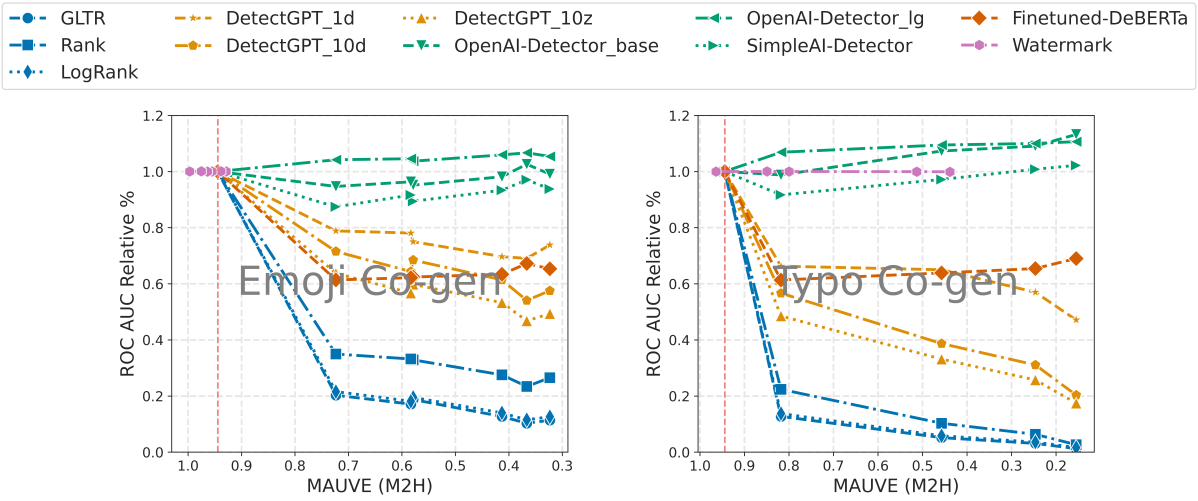

Figure 18: Performance drop under the co-generating attacks with **MAUVE** (M2H) as budget (x-axis).

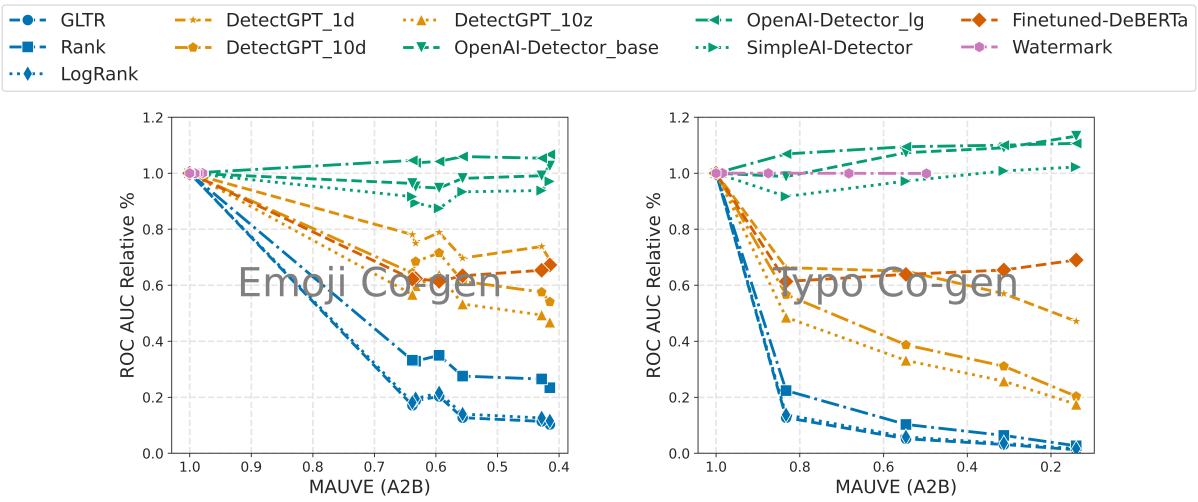

Figure 19: Performance drop under the co-generating attacks with **MAUVE** (A2B) as budget (x-axis).

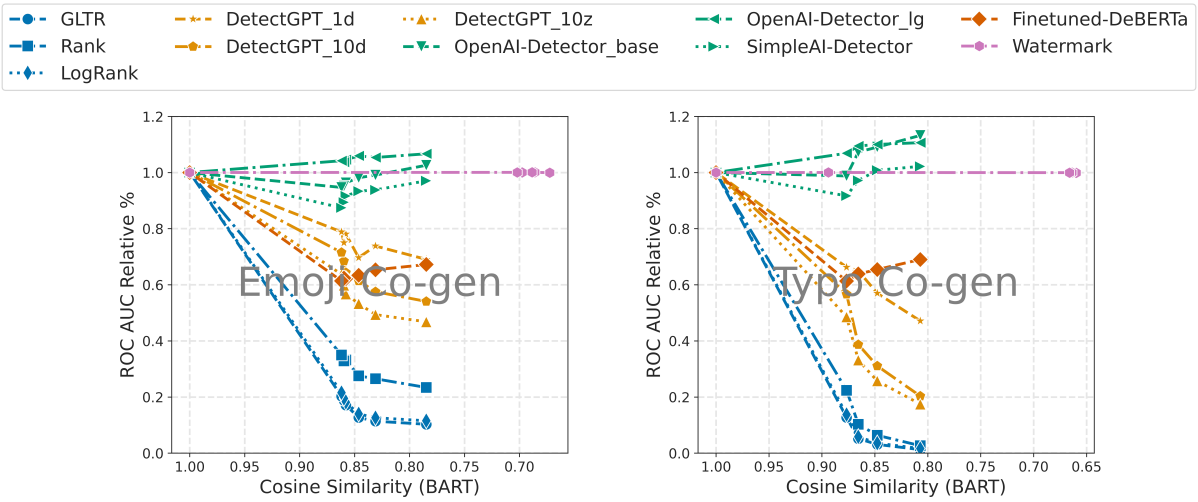

Figure 20: Performance drop under the co-generating attacks with **Cosine Similarity** as budget (x-axis).

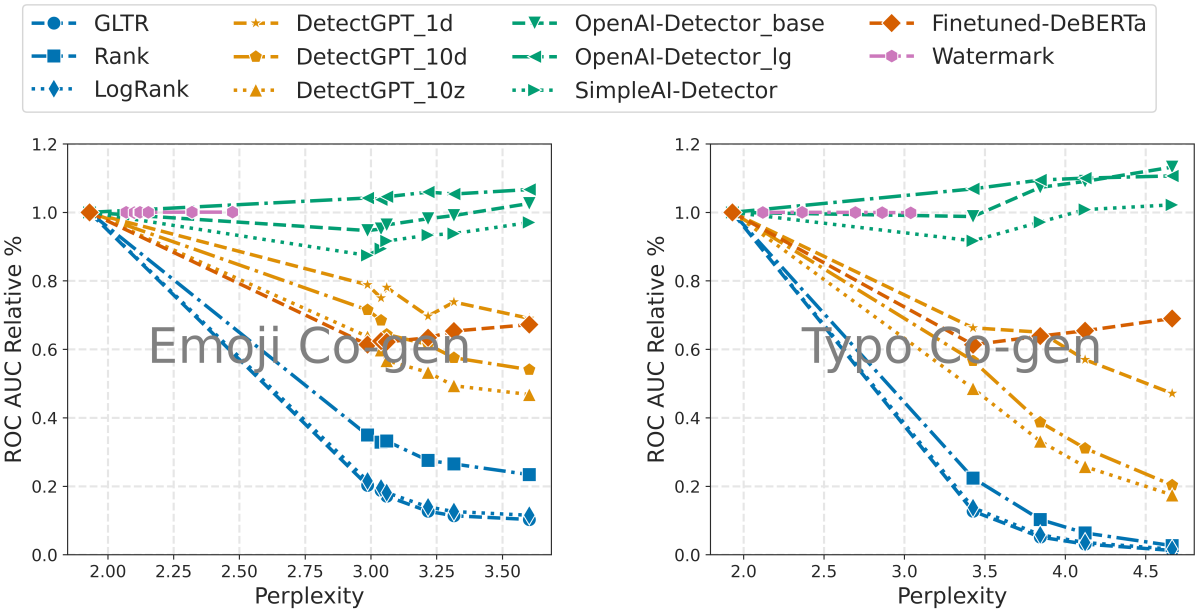

Figure 21: Performance drop under the co-generating attacks with **Perplexity** as budget (x-axis).

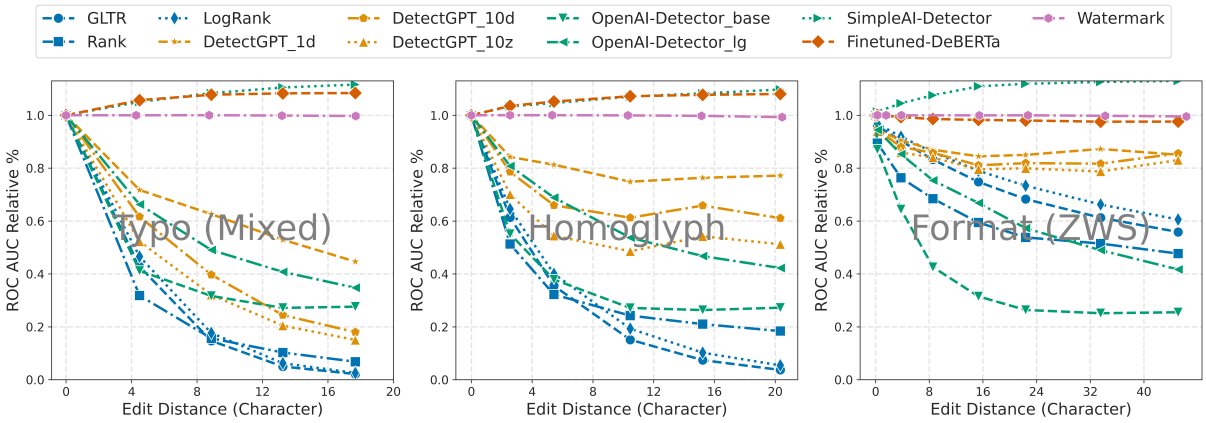

Figure 22: Performance drop under the editing attacks with relative **ROC AUC** as performance metrics (y-axis).

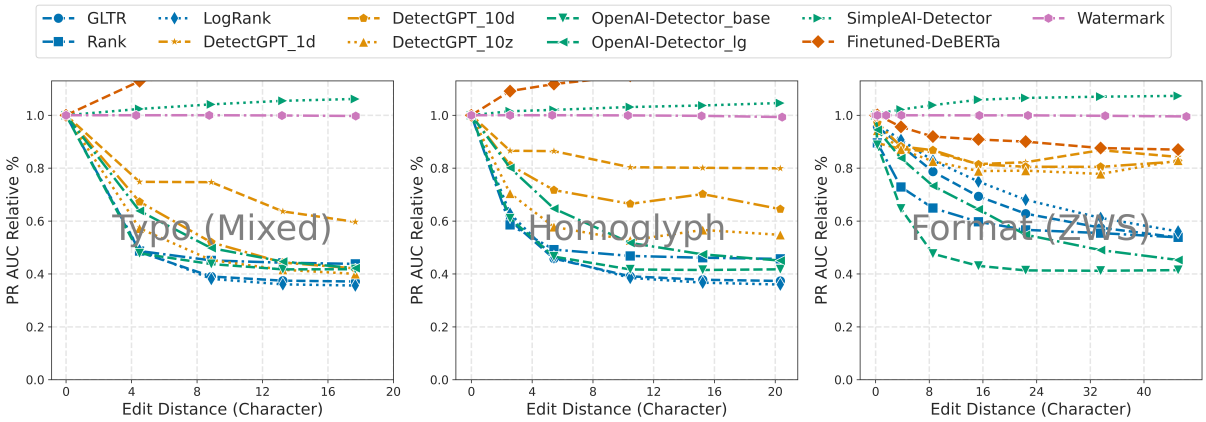

Figure 23: Performance drop under the editing attacks with relative **PR AUC** as performance metrics (y-axis).

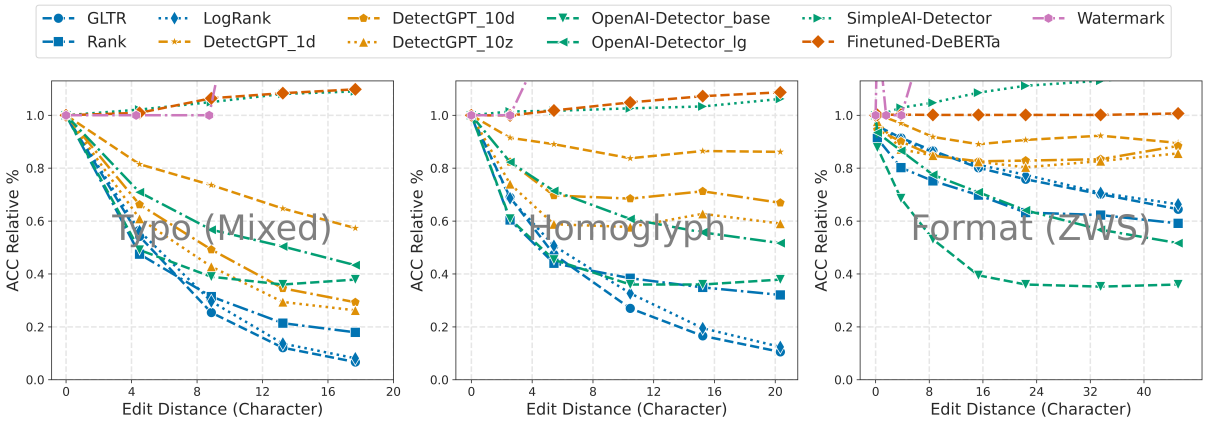

Figure 24: Performance drop under the editing attacks with relative **accuracy** as performance metrics (y-axis).

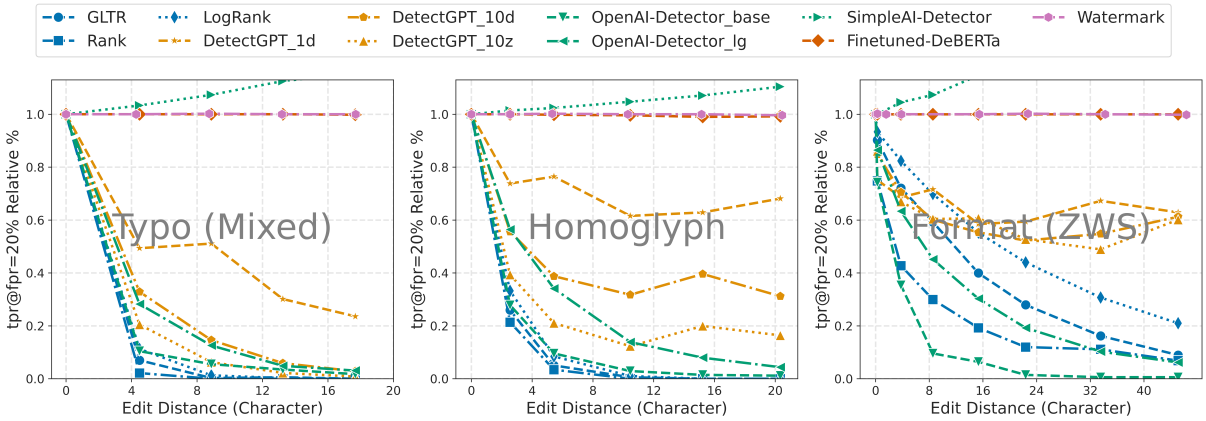

Figure 25: Performance drop under the editing attacks with relative **TPR@FPR=20%** as performance metrics (y-axis).

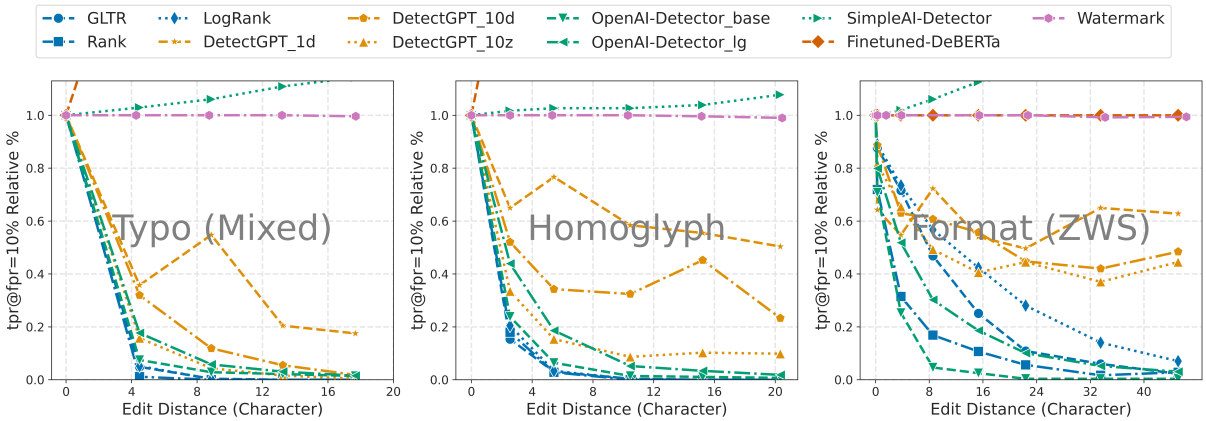

Figure 26: Performance drop under the editing attacks with relative **TPR@FPR=10%** as performance metrics (y-axis).

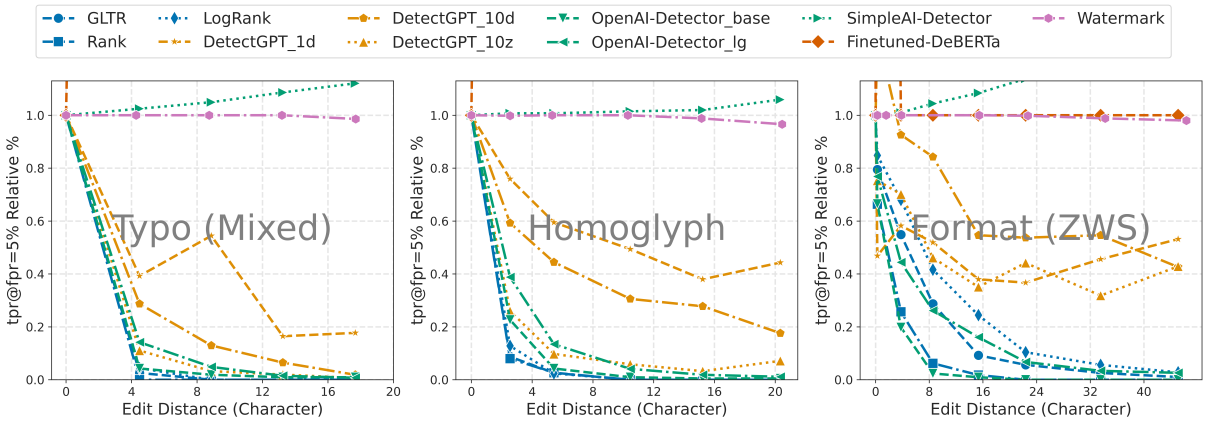

Figure 27: Performance drop under the editing attacks with relative **TPR@FPR=5%** as performance metrics (y-axis).

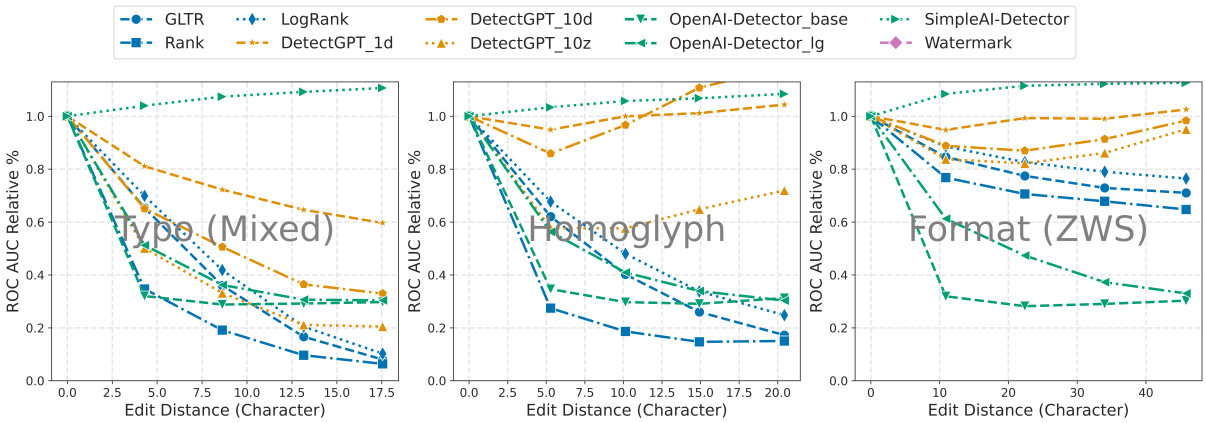

Figure 28: **LlaMA-2 dataset**: Performance drop under the editing attacks with Edit Distance as budget (x-axis).

Prompt	**A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio**
HWT	A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio and newspapers combined, and Internet reports are considered much more reliable that other media. It also shows only 1 in 200 people surveyed believes newspapers will be a dominant source of information in 2014. The survey discovered 56 percent of adults nationwide would pick the Internet if they were allowed just one source for their news, while television, newspapers and radio earned the support of 41 percent – together. Among Republicans, 56 percent would choose the Internet for their news, while among Democrats that figure was 50 percent.
unattacked MGT	A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio or newspapers. And it's true: only a third of Americans get their news from newspapers, TV or radio. In a new Pew Research Center poll, only 37 percent of Americans said they get their news from newspapers. That's down from 44 percent who said they got news from newspapers in 2010. TV news was a distant second in the poll. Only 23 percent of Americans said they get their news from TV. That's down from 26 percent in 2010.
unattacked watermarked MGT	A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio, newspapers or magazines. The poll, conducted by the News Literacy Project, found that 42 percent of American adults say they rely exclusively on the Internet for news. That's up from 33 percent in 2008. At the same time, the number of people relying on television for information has declined since the poll began asking this question in 2003. In 2003, 47 percent of Americans said they used TV exclusively for news. That's down to 38 percent in 2008 and 34 percent today.
typo insertion (mixed)	*Description*: Create typos by a mixture method of inserting, deleting, substituting, and transposing. *Config*: Insert typo in 20% of tokens, one edit per selected token. A new poll reveals that more Americans would choose the Internet as their only news source than TV, rapio or newsbapers. And it's true: only a third of Ameyicans get their news from newspapers, TV or radio. In a now Pew Research Center poll, only 37 percent of Americans said they ges their nws fro newspapers. Bhat's down rrom 4 percent who aid hthey got nelws from newspapers in 2010. TV news wae a distant secod in the poll. Only 23 percent of Americans said hey get their news fdom TV gThat's down from 26 percent in 2010.
homoglyph alternation	*Description*: Change English characters into visually similar Unicodes. *Config*: Change 20% of tokens, one homoglyph character per selected token. A new poll reveals thaτ more Americans would choose the Internêt as their only news source than TV radio or newspapers. And it's true: oňly a third of Americans get their news from newspapers, TV or radio. In a new Pew Research Ceňter poll, only percent of Americans said they get their neωs from newspapers. Thât's down from 44 percent who said they gôt news from newspapers in 2010. TV newŝ was a distant second în the poll. Only 23 peřcent of Amêricans said they get their nêws from TV. Thaτ's down from 26 percenτ in 2010.
format character editing (zero-width whitespace)	*Description*: Insert formatting characters, i.e., zero-width whitespace \u200B. *Config*: Insert \u200B at the end of 20% of tokens. A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio or newspapers. And it's true: only a third of Americans get their news from newspapers, TV or radio. In a new Pew Research Center poll, only 37 percent of Americans said they get their news from newspapers. That's down from 44 percent who said they got news from newspapers in 2010. TV news was a distant second in the poll. Only 23 percent of Americans said they get their news from TV. That's down from 26 percent in 2010. [*Note: zero-width whitespace can not be displaced in LaTeX.*]
synonyms substitution model-free	*Description*: Retrieve synonym from a static dictionary to substitute. *Config*: Substitute 10% words. A new poll reveals that more American English would choose the Internet as their only news source than TV, radio, newsprint or magazines. The crown, conducted by the news show Literacy Project, found that 42 percent of American English adults say they rely exclusively on the Internet for news. That's up from 33 percent in 2008. At the same clock, the bit of people relying on television for info has declined since the poll began asking this interrogative sentence in 2003. In 2003, 47 percent of American English said they used TV exclusively for news show. That's down to 38 percent in 2008 and 34 percent today.

synonyms substitution model-based	*Description*: Utilize an LLM to generate a synonym list given context. *Config*: Substitute 10% words using LlaMA with temperature 1. `A new poll reveals that more Americans would choose the Internet as their only news source than TV, depend on solely newspapers or magazines. The poll, conducted by the News Literacy Project, found that 42 percent of American adults say they rely exclusively on radio Internet for news. That's survey from 33 percent in 2008. At the same time, the number of people relying on television solely information has declined since the poll began asking this question in 2003. In yesterday, 47 percent of Americans said they used TV exclusively for news. That's down to 38 percent in 2008 and 34 percent today.`
span perturbation	*Description*: Use a masked LM (Raffel et al., 2020) to rewrite spans of tokens by masked filling. *Config*: Rewrite 30% of 2-spans. `A new poll reveals that more people choose the Internet as their primary news source than TV, radio, newspapers or magazines. The survey, conducted by The Media Literacy Project, found 73 percent of American poll respondents said they rely exclusively on the Internet for news. It's up from 33 percent in 2008. At the same time, the number of people relying on television for news has declined since the poll began asking questions in 2007. In 2003, 47 percent of Americans said they used TV exclusively to get their news. That's down to 38 percent in 2009, and 34 percent today.`
inner-sentence paraphrasing	*Description*: Use Pegasus (Zhang et al., 2020) to paraphrase each sentence of the text and then join them. *Config*: Paraphrase 80% of the sentences. `A new poll shows that more Americans would prefer the internet to TV, radio, newspapers or magazines. The poll, conducted by the News Literacy Project, found that 42 percent of American adults say they rely exclusively on the Internet for news. It was up from 33 percent in 2008. At the same time, the number of people relying on television for information has declined since the poll began asking this question in 2003. Almost half of Americans used TV exclusively for news in 2004. In 2008 it was 38 percent and today it is 34 percent.`
inter-sentence paraphrasing	*Description*: Paraphrase with Dipper (Krishna et al., 2023), a paragraph-level paraphraser that can re-order, split, and merge sentences. *Config*: Lexical (L) diversity 60 and Order (O) diversity 60. `Most Americans get their news from television, radio, and old-fashioned newspapers. In a recent poll by the Pew Research Center only 37 percent reported that they get news from newspapers. And that's true. More Americans would now use the Internet to get their news than would do so newspapers, radio, or the Americanized news service. In the survey, "American View" was the second most favored channel, with 23 percent, and the Americanized news service was the third most favored with 13 percent. In 2010, 44 percent of Americans said they would get their news from newspapers. Now that's just 33 percent. Down from the old-fashioned 32 percent in 2010.`
prompt paraphrasing	*Description*: Paraphrase the raw prompt before generation using Pegasus. *Config*: - `More Americans would choose the internet as their only news source, according to a new poll released recently. The survey shows a significant shift in how people consume news, with a growing preference for digital platforms over traditional media like television or newspapers. Respondents cited the convenience, immediacy, and accessibility of online sources as key reasons for this shift. However, concerns about misinformation and fake news persist, emphasizing the need for diligent fact-checking and critical evaluation of digital content. The study underscores the significant influence of the internet on shaping public opinion and the rapidly changing media landscape.`
in-context learning	*Description*: Given the example of HWT and MGT as positive and negative demonstrations when generating MGT on the same prompt. *Config*: 1 positive demonstration + 1 negative demonstration. `A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio, and print media. The survey further indicates that 60% of the respondents rely on social media for daily news updates, highlighting the increasing importance of platforms like Facebook, Twitter, and Instagram in news dissemination. The trend is particularly prominent among the younger demographic, with 70% of those aged 18-34 choosing the internet as their primary news source. Despite this, concerns about the spread of fake news and misinformation online persist, stressing the need for digital literacy and fact-checking mechanisms.`

character-substituted generation	*Description*: Prompt to ask the model to generate the text with specific character substitution criteria and recover the output after finishing the whole generation. *Config*: Substitute 'c's and 'k's using GPT-4. `A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio or newspapers. This shift in preferenae highlights the growing influence of the Internet and digital technologa in shaping public opinion. It's not surpriaing, given the ezae of access to information online. Yet, some raise concern over the increasing relianae on the web for news, as it makes it easier to spread misinformation. With fake news becoming a common issue, there's a growing need for digital literacy and critical thinking skills to discern fact from fiction.`
typo co-generation	*Description*: Preset character substitution rules and execute the rules when finishing sampling each token and recover them after finishing the whole text generation. *Config*: Substitute 'c's and 'k's with 50% probability. `A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio or newspapers. The results 73 percent are reported by Pew Research that were conducted on an average of views based on discussions and polls around the Internet, and express popular ingestion vivid facts. The majority of Americans choose fer the Internet as their source of faksual information rather than TV news or radio news, reports Pew. About half online respondents (52%), for example, say they go there when they need information they likely folllow upwere.`
emoji co-generation	*Description*: Compulsorily generate or insert an emoji after finishing each sentence while recurrent generation and remove all the emojis after finishing the whole text. *Config*: Insert 1 emoji at the end of 20% sentence. `A new poll reveals that more Americans would choose the Internet as their only news source than TV, radio or any print publication. A new study by social analytics firm Altimeter Group confirms what has been obvious to most consumers since 2017 We are a far more information-driven species, and are blown away with the freely available, POV-polling and influence-mongering power of the Internet perspective Quinnipiac University Polls conducted during April and May 2018 reveal that an astounding eight in ten voters say they trust internet polls in deciding important battles, versus an abysmal 49 in 2016.`

Table 15: **Example of the attacks with description and configs.**

Metric		*Scale*	*Definition*
Levenshtein Edit Distance (Levenshtein, 1965)		$\geq 0 \uparrow$	The minimum number of single-character edits (insertions, deletions or substitutions).
Jaro Similarity (Jaro, 1989)		$\geq 0 \downarrow$	A similarity metric based on matching characters and transpositions in two strings.
Perplexity (PPL)		$> 0 \leftrightarrow$	Apply Llama-7B-hf (Touvron et al., 2023b).
MAUVE (Pillutla et al., 2021)	M2H	$(0, 1] \leftrightarrow$	MGTs to estimate the model distribution Q and HWTs to estimate the target distribution P. For attacked scenarios, the closer value to the unattacked scenario is favored.
	A2B	$(0, 1] \downarrow$	MGTs (attacked) to estimate the model distribution Q and MGTs (unattacked) to estimate the target distribution P.
Cosine Similarity		$[-1, 1] \downarrow$	Utilize BART embedding (Lewis et al., 2020) to compare the similarity of texts after the attack to before the attack.
BERTScore (Zhang et al., 2019)	M2H	$[0, 1] \leftrightarrow$	MGTs as the candidates $\hat{x}$ and HWTs as the reference x. For attacked scenarios, the closer value to the unattacked scenario is favored.
	A2B	$[0, 1] \downarrow$	MGTs (attacked) as the candidates $\hat{x}$ and MGTs (unattacked) as the reference x.
BARTScore (Yuan et al., 2021)	M2H	$< 0 \leftrightarrow$	MGTs as the source x and HWTs as the target y. For attacked scenarios, the closer value to the unattacked scenario is favored.
	A2B	$< 0 \downarrow$	MGTs (attacked) as the source x and MGTs (unattacked) as the target y.
Semantics Human Eval		$[0, 1] \downarrow$	Pairing attacked MGTs with the unattacked, asking humans to judge whether they are semantic-similar.
Quality Human Eval		$[0, 1] \downarrow$	Pairing attacked MGTs with the unattacked, asking humans to judge which one is more high-quality.

Table 16: **The metrics considered to evaluate the budget of attacks.** $\uparrow$ means a larger number represents a more significant attack on the raw texts. $\leftrightarrow$ means the value closer to the value of unattacked texts is favorable. 'M2H' is 'MGT to HWT,' and 'A2B' is 'After to Before Attack' for short. Metrics in grey are not distinguishable enough empirically that we do not show in the paper, but are also implemented and reported in our code and data repertory.