Efficient Black-Box Adversarial Attacks on Neural Text Detectors

Vitalii Fishchuk
University of Twente
Faculty of Electrical Engineering,
Mathematics and Computer Science
v.fishchuk@student.utwente.nl

Daniel Braun
University of Twente
Department of High-tech Business
and Entrepreneurship
d.braun@utwente.nl

Abstract

Neural text detectors are models trained to detect whether a given text was generated by a language model or written by a human. In this paper, we investigate three simple and resource-efficient strategies (parameter tweaking, prompt engineering, and character-level mutations) to alter texts generated by GPT-3.5 that are unsuspicious or unnoticeable for humans but cause misclassification by neural text detectors. The results show that especially parameter tweaking and character-level mutations are effective strategies.

1 Introduction

The widespread availability of neural text generation models, like ChatGPT, has caused an increased desire for neural text detectors, i.e. models that can detect whether a given text was AI-generated. The reliance of, e.g., educational institutions on such detectors has raised questions about their robustness in general and in specific with regard to adversarial attacks (Jawahar et al., 2020; Wolff and Wolff, 2022; Liang et al., 2023a,b). Such attacks exploit the fact that machine learning models by identifying patterns in the data rather than by understanding actual underlying concepts. Consequently, introducing small, human-unnoticeable perturbations can result in misclassification. (Goodfellow et al., 2014; Szegedy et al., 2013)

Adversarial attacks can be categorised into black-box and white-box attacks (Peng et al., 2023). In white-box attacks, the attacker has full access to the target model, including its parameters, architecture, and loss function (Ebrahimi et al., 2018; Gao et al., 2018). During black-box attacks, the adversary can only input queries and observe the outputs without any insights into internal processing (Gao et al., 2018). Furthermore, it can be distinguished between targeted and untargeted attacks, where targeted attacks aim at triggering misclassification towards a specific label, while untargeted aim to cause any misclassification (Rathore et al., 2021).

This paper investigates effective and resource-efficient universal attack strategies in a black-box scenario with minimal resources, based on text generated with GPT 3.5 and three neural text detectors: the widely used open source GPT-2 Output detector model[1], the OpenAI text classifier[2], and the commercial Turnitin AI detector[3], which is used by many educational institutions. The results show that character-level mutations, tweaking the parameters of the generative model, as well as prompt engineering, are efficient and effective strategies, showing that currently available neural text detectors can not reliably detect texts generated by state-of-the-art large language models (LLMs).

2 Related work

Most of the existing literature about adversarial attacks focuses on image detection. Textual input is less used due to its discrete nature and the difficulty in introducing human-imperceptible perturbations, contrary to the image data, where a change in a few hundred pixels can go unnoticed (Jin et al., 2019; Peng et al., 2023). Examples of adversarial attacks on general text classification models include the work by Ebrahimi et al. (2018) and Gao et al. (2018). More recent work has started to specifically look into adversarial attacks on neural text detectors: Wolff and Wolff (2022) showed that introducing spelling mistakes and replacing characters with homoglyphs can significantly reduce the detection rate for GPT-2 texts. Liang et al. (2023a) showed that similar character-level mutation-based attacks are also successful for RoBERTa-based detection models. Liang et al. (2023c) not only showed that

[1]https://github.com/openai/gpt-2-output-dataset/tree/master/detector
[2]https://platform.openai.com/ai-text-classifier
[3]https://www.turnitin.com

existing detectors are vulnerable to simple rephrasing, but they also showed that they are biased towards flagging texts that have been (manually) written by non-native speakers as AI-generated.

Because currently available methods are vulnerable to adversarial attacks, multiple suggestions have been made to improve their robustness, e.g. by Liang et al. (2023b), Shen et al. (2023), Crothers et al. (2022), and Yoo et al. (2022). While watermarking techniques to identify AI-generated texts are also investigated, they are generally seen as vulnerable to adversarial attacks, especially to mutation and paraphrasing-based approaches (Jin et al., 2019; Kirchenbauer et al., 2023; Sadasivan et al., 2023).

3 Approaches

Based on the existing literature, we identified three promising and efficient approaches for adversarial attacks: parameter tweaking, prompt engineering and character-level mutations. All approaches were tested with the three neural text detectors mentioned in Section 1: GPT-2 output detector, OpenAI classifier, and Turnitin AI writing detector. The basis for all attacks were texts generated by the GPT-3.5-turbo model via the OpenAI API. The text samples were produced as 500-word essays, with topics taken from a list of 200 essay topics (Nova, 2019) and the prompt *"Write a five-hundred-word argumentative essay on the topic 'topic'."*. It was then evaluated how the detection rate changed between the original texts and their altered version. To ensure comparability of the results between different detectors, all scores were projected onto a scale from 0.0 (very likely not AI-generated) to 1.0 (very likely AI-generated). The GPT-2 Output detector returns a score between 0.0 and 1.0, that can be used directly. Turnitin returns a percentage between 0 and 100 indicating how much of the text was generated by AI. We divide the score by 100. The OpenAI classifier returns one of five labels ("very unlikely", "unlikely", "unclear", "possibly", "likely"). For each of the labels, OpenAI (2023a) provides a corresponding range of numerical thresholds, of which we take the mean score (0.05, 0.275, 0.675, 0.94, 0.99). The code for the evaluation was written with the assistance of GPT-4, followed by extensive testing of the code, as well as additional, manually implemented, features. The code and the data are available on GitHub[4].

Parameter	Min	Max	Default
Temperature	0.0	2.0	1.0
Top p	0.0	1.0	1.0
Frequency penalty	-2.0	2.0	0.0
Presence penalty	-2.0	2.0	0.0

Table 1: Investigated parameters

3.1 Parameter tweaking

First, we investigated the influence of GPT-3.5 generation parameters on the detection. Table 1 shows the parameters we focus on because they have the biggest impact on the produced texts according to OpenAI (2023b). Temperature and top p control randomness in the text. By increasing the temperature, the output becomes more random. However, for values beyond the default of 1.0, the length of the outputs started fluctuating strongly and the quality of the texts dropped. Consequently, we focused on the range between 0.0 and 1.0. Top p represents the percentage of tokens selected based on their probability mass. The frequency penalty controls the frequency of tokens appearing in the text, with higher values leading to more diverse verbatim. During the testing phase, it was found that increasing the frequency penalty beyond 1.0 degrades the quality of the texts rapidly. Additionally, as decreasing the value below 0.0 increases the repetitiveness, we focused on the range between 0.0 and 1.0. Finally, the presence penalty controls the model's likelihood of repeating tokens in the text. Higher values of presence penalty lead to the model producing more diverse texts. Following consideration similar to the frequency penalty, the negative values were discarded, and we focused on the range between 0.0 and 2.0. (OpenAI, 2023b) First, we investigated each parameter separately, changing it in steps of 0.1. Subsequently, we used the two parameters that had the biggest impact and investigated whether their interaction also influences the detection rate by performing a grid search in steps of 0.1 with these parameters.

3.2 Prompt engineering

The second approach explored the effect of prompt engineering on detection rates. Since Liang et al. (2023c) have shown that detection is vulnerable towards simple rephrasing, our hypothesis was that

[4]`https://github.com/Lolya-cloud/adversarial-attacks-on-neural-text-detectors`

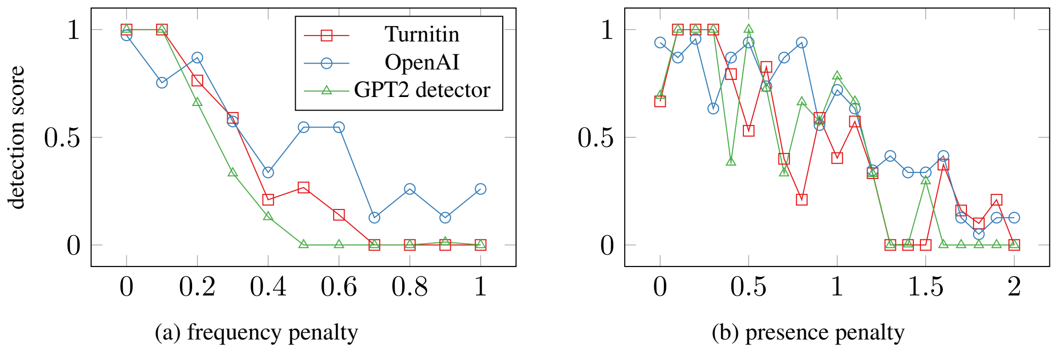

Figure 1: Influence of the frequency and presence penalty on the detection score

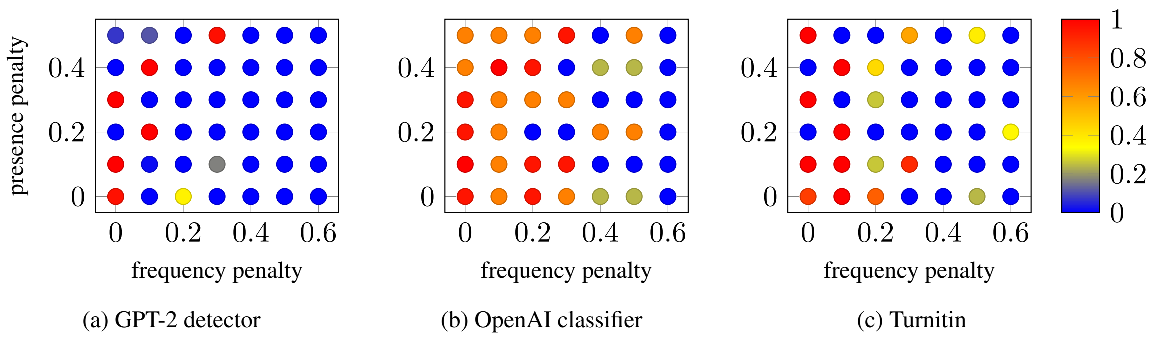

Figure 2: Influence of joined optimsation of fequency and presence penalties on the detection score

providing regeneration instructions as a separate prompt might lower the detection rate. This hypothesis was tested with the following four prompts, each of which was used to generate ten texts: (1) standard prompt as described in Section; (2) standard prompt followed by second query *"Regenerate the essay."*; (3) advanced prompt as shown in Appendix A.1; (4) standard prompt followed by advanced prompt as second query. Additionally, several prompts to increase the perplexity and burstiness of generated texts were tested, a strategy that is popular among users (see e.g. Alexander (2023)). Ten prompts were designed, and for each, ten texts were generated. The first five prompts use single-query architecture, while the others utilise a two-query concept. The following methods were tested with different prompts: (1) Explaining burstiness and perplexity, then asking to implement them (see Appendix A.2); (2) Explicitly asking to maximise either perplexity, burstiness or both (standard prompt followed by *"Maximize the burstiness / perplexity / burstiness and perplexity of the text."*); (3) Explicitly asking to rewrite to avoid detection. (*"Rewrite the above essay in order to avoid AI detection."*)

3.3 Character-level mutations

This approach aimed to test the robustness of the detectors against traditional adversarial attack vectors. Three character level mutations were taken as a basis: replacing either Latin lowercase "a" or "e" with the corresponding Cyrillic analogue; replacing Latin lowercase "L" with Latin uppercase "I". Ten texts were generated with the standard prompt and then the mutations were applied.

4 Results

4.1 Parameter tweaking

The detection rate of all three detectors dropped with an increment in either frequency or presence penalty (see Figure 1). Starting from a frequency penalty of 0.3-0.4 and a presence penalty of 1.0-1.2, the detection rate fell under 50% with some fluctuations. An analysis of the generated texts showed that increasing either the frequency or presence penalty led to more diverse texts. A higher frequency penalty caused a wider vocabulary variety. For values above 0.6, the occurrence of punctuation mistakes and unclear wordings quickly increased, making the texts difficult to read. For values between 0.0 and 0.6, increases in value caused an incremental increase in text complexity while preserving quality and readability. A higher presence penalty primarily influenced the diversity of perspectives and text engagement. However, for values above 0.6, the coherence and logical progression rapidly decreased and texts became less subjectfocused. Therefore, increasing either frequency or presence penalty from the default of 0.0 up to 0.6 can be seen as successful attack strategies that significantly lowers detection while maintaining text quality. Increasing the temperature and top p value above the default was already excluded from the experimental design, because of the strong negative effects on text quality. Lowering either value resulted in more deterministic outputs leading to higher detection rates, therefore, tweaking those parameters is not a successful attack strategy.

In a second step, the interaction between frequency and presence penalty was investigated. Figure 2 shows that the detection rates dropped for all three detectors with an increment in frequency and presence penalty. Notably, they started dropping for smaller values of presence and frequency penalty than they did for the individual parameters, thereby minimising the potential negative effects on text quality. The GPT-2 detector showed the worst performance in the comparison with very quickly declining detection scores.

4.2 Prompt engineering

The simple regeneration approach, whether using a second query or providing detailed instruction in the first query, did not have an impact on detection rates. Increasing the perplexity and burstiness of the texts through prompts, on the other hand, caused a drop in the detection rate across all three detectors, however only if applied in two separate prompts, as shown in Figure 3. Although the approach managed to decrease the detection rate across all three detectors, the score sank only for the GPT-2 detector below 0.5 (see Figure 3a). For the other two detectors, the score stayed above 0.5, meaning that the generated texts would still be detected as AI-generated or at least as undecidable.

4.3 Character-level mutations

The influence of the character-level mutations on the detection scores is shown in Table 2. For the GPT-2 detector, all three replacements lead to a detection score of 0. For Open AI, all attacks lowered the detection score, although not as much as for the GPT-2 detector. Turnitin detected the replacement of Latin characters with Cyrillic characters and flagged the attack. The substitution of *l*s with capital *i*s, however, remained undetected and significantly lowered the detection score, therefore presenting a successful attack strategy.

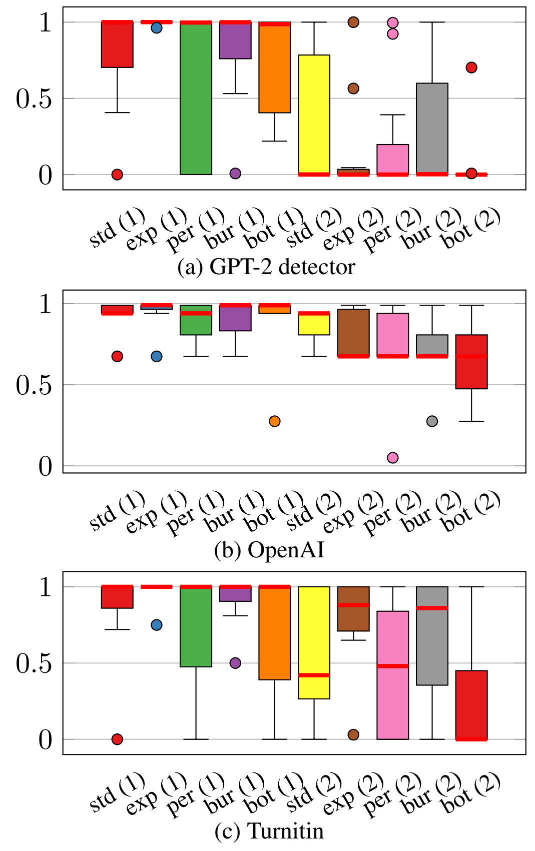

Figure 3: Influence of perplexity and burstiness prompts on detection (number of prompts in brackets; std = standard prompt; exp = prompt to explain and increase burstiness & perplexity; per = increase perplexity; bur = increase burstiness; bot = increase both)

5 Conclusion

This study explored the effectiveness of resource-efficient adversarial attacks on neural text detectors, based on texts generated by GPT-3.5 and the three detectors GPT-2 output detector, OpenAI classifier, and Turnitin. Of the three investigated strategies, parameter tweaking and character-level mutations were successful for all three detectors. Prompt engineering was only successful for the GPT-2 output detector. All strategies are resource efficient and easy to implement, effectively showing that currently available detectors cannot reliably detect AI-generated texts and are vulnerable to adversarial attacks.

	GPT-2	OpenAI	Turnitin
Standard	0.67	0.77	0.75
Swap a lat.-cyr.	0	0.52	x
Swap e lat.-cyr.	0	0.48	x
Swap l - I	0	0.38	0.21

Table 2: Character-level mutation mean detection score

References

Chris Alexander. 2023. Asking chatgpt to put perplexity and burstiness in an essay appears to fool ai detectors. https://telblog.unic.ac.cy/teaching-chatgpt-perplexity-burstiness-appears-to-fool-ai-detectors/. Last accessed: 2023-07-11.

Evan Crothers, Nathalie Japkowicz, Herna Viktor, and Paula Branco. 2022. Adversarial robustness of neural-statistical features in detection of generative transformers. *Proceedings of the International Joint Conference on Neural Networks*, 2022-July.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018*, pages 50–56.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 8018–8025.

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

Gongbo Liang, Jesus Guerrero, and Izzat Alsmadi. 2023a. Mutation-based adversarial attacks on neural text detectors.

Gongbo Liang, Jesus Guerrero, Fengbo Zheng, and Izzat Alsmadi. 2023b. Enhancing neural text detector robustness with &mu;attacking and rr-training. *Electronics*, 12(8).

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023c. GPT detectors are biased against non-native english writers. In *ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*.

A. Nova. 2019. Essay topics: 100+ best essay topics for your guidance. https://www.5staressays.com/blog/essay-writing-guide/essay-topics. Last accessed: 2023-07-11.

OpenAI. 2023a. Ai text classifier - openai api. https://platform.openai.com/ai-text-classifier. Last accessed: 2023-07-11.

OpenAI. 2023b. Api reference - openai api.

Hao Peng, Zhe Wang, Dandan Zhao, Yiming Wu, Jianming Han, Shixin Guo, Shouling Ji, and Ming Zhong. 2023. Efficient text-based evolution algorithm to hard-label adversarial attacks on text. *Journal of King Saud University - Computer and Information Sciences*, 35:101539.

Pradeep Rathore, Arghya Basak, Sri Harsha Nistala, and Venkataramana Runkana. 2021. Untargeted, targeted and universal adversarial attacks and defenses on time series. *Proceedings of the International Joint Conference on Neural Networks*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected?

Lujia Shen, Xuhong Zhang, Shouling Ji, Yuwen Pu, Chunpeng Ge, Xing Yang, and Yanghe Feng. 2023. Textdefense: Adversarial text detection based on word importance entropy.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*.

Max Wolff and Stuart Wolff. 2022. Attacking neural text detectors.

Ki Yoon Yoo, Jangho Kim, Jiho Jang, and Nojun Kwak. 2022. Detection of word adversarial examples in text classification: Benchmark and baseline via robust density estimation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 3656–3672.

A Appendix - Prompts

A.1 Advanced Prompt

"Write a five-hundred-word argumentative essay on the topic 'topic'. Include personal reflections, use a mix of long and short sentences, employ rhetorical questions to engage the reader, maintain a conversational tone in parts, and play around with the paragraph structure to create a dynamic and engaging piece of writing. Try to include factual and contextual information, use advanced concepts and vocabulary. Utilize a combination of complex and simple vocabulary. Try to mimic human writing as closely as you can. Avoid passive voice, as it tends to occur more often in AI-generated texts. Add a few examples from the real world illustrating your point."

A.2 Perplexity and Burstiness

"Write a five-hundred-word argumentative essay on the topic 'topic'. When it comes to writing content, two factors are crucial, perplexity and burstiness. Perplexity measures the complexity of the text. Separately, burstiness compares the variations of sentences. Humans tend to write with greater burstiness, for example, with some longer or more complex sentences alongside shorter ones. AI sentences tend to be more uniform. Therefore, when writing the following content, I need it to have a good amount of perplexity and burstiness.