Adversarial attacks on neural text detectors

VITALII FISHCHUK, University of Twente, The Netherlands

This thesis explores the effectiveness of adversarial attack methods in evading AI-text detection. Experimenting on three attack categories, prompt engineering, parameter tweaking, and character-level mutations, this research employs a mixed-methods approach to examine the effectiveness of such attacks with the recently released GPT-3.5 model. Results from this research reveal the low robustness of existing detectors towards practical and resource-efficient attack methods. The findings demonstrate how prompt engineering, parameter tweaking and character-level mutations can be exploited to evade detection effectively. Additionally, the study shows that detector algorithms struggle with the GPT-4 model and highlights the need for urgent improvement in existing detectors.

Additional Key Words and Phrases: Natural language processing (NLP), Adversarial attacks, GPT-3.5

1 INTRODUCTION

Recent advancements in deep learning resulted in significant improvements in Natural Language Processing (NLP), with GPT-3, GPT-3.5, and GPT-4 showing exceptional performance in language understanding and text generation. Concurrently, AI-generated text detectors (Models that are trained to detect whether a text is AI-generated) also became increasingly sophisticated. Because of the widespread deployment of these NLP models, the detection of AI-generated texts is becoming more important. AI-generated text detectors have become increasingly sophisticated, making them an interesting target for adversarial attacks.

Such attacks exploit the fact that any Machine Learning (ML) model works by identifying patterns in the data rather than by understanding actual underlying concepts [7, 23]. Consequently, introducing even tiny human-unnoticeable perturbations to the sample could result in misclassification by the model [7, 23]. These modified inputs are typically referred to as adversarial examples.

Adversarial attacks can be categorised into black-box and white-box attacks [19]. In white-box adversarial attacks, the attacker has full access to the target model, including its parameters, architecture and loss function [5, 6]. In contrast, during the black-box attack, the adversary can only input queries and observe the outputs without any insights into internal processing [6].

Furthermore, adversarial attacks can be classified into targeted and untargeted attacks, where targeted attacks aim at triggering misclassification towards a specific label, while untargeted aim to cause any misclassification. A further distinction can be made between individual and universal adversarial attacks, with the former aiming at finding unique perturbations for each input sample and the latter - at finding perturbation patterns applicable to all samples in the dataset. [20]

TScIT 39, July 7, 2023, Enschede, The Netherlands

Although there is much literature on adversarial attacks on image detection, the text domain is less explored due to its discrete nature [8, 19]. This discrepancy also results from the difficulty in introducing human-imperceptible perturbations to text, contrary to the image data, where a change in a few hundred pixels can go unnoticed [19]. Although recent studies have begun to address this gap [10, 22], the most recent generation of models is yet to be investigated.

Several black-box adversarial attack methods exist. Still, the majority of them fall under the following categorisation:

- **Surrogate model attacks**: A simple surrogate model is trained locally to craft adversarial examples for the target model. This technique approaches a black-box environment as a white box with some limitations. [12]
- **Transfer-based attacks**: Adversarial examples are crafted on a different available classification model and then transferred to the target model. [2, 27]
- **Query-based attacks**: Adversarial examples are produced by repeatedly querying the target model and adapting based on the feedback received. Produces significant model overhead. [3, 4, 8, 10, 25]

The first two methods assume the transferability of adversarial attacks between models. The only difference between them is whether a new model is trained. Contrarily, the third method works directly with the target model, potentially leading to a higher success rate but producing significant model overhead.

With the arrival of ChatGPT, prompt engineering has emerged as a potential new method for adversarial attacks. In the context of NLP, prompt engineering refers to crafting model inputs to customise interaction and outputs [24].

This research explores prompt engineering, GPT 3.5 parameter tweaking and character-level mutations for creating targeted adversarial attacks against widely-used detectors in a black-box scenario with minimal resources. The ultimate goal is to assess whether an effective and resource-efficient universal attack strategy can be developed in this scenario.

2 RESEARCH QUESTION

During this study, the following research question is answered: *What adversarial attack methods can successfully bypass AI-generated text detectors in a black-box, limited-resource environment on the GPT-3.5 model?* Various experiments were conducted to create adversarial samples from GPT-3.5-generated texts. The adversarial samples were fed to the Turnitin AI detector[1], OpenAI classifier[2] and GPT-2 Output detector[3]. The attack strategies were assessed to answer the research question and draw conclusions.

[1] https://www.turnitin.com/solutions/ai-writing
[2] https://platform.openai.com/ai-text-classifier
[3] https://openai-openai-detector--jsq2m.hf.space/

3 LITERATURE REVIEW

Although the literature on adversarial attacks on NLP models is limited, a larger volume of research exists within a similar field of image recognition and computer vision, which provides an important reference point. However, their applicability is hindered due to the discrete nature of the text domain and the difficulty in introducing human-imperceptible perturbations.

Several studies operating in an image black-box environment explore the concepts of surrogate models and transfer-based attacks [2, 12, 27]. While using different methods, they all require estimating the target model loss function to compute the gradient. This requirement contrasts the strategies proposed in the research on query-based image attacks [3, 4] and text attacks [8, 10, 25], which operate independently of the loss function, thereby enhancing resource efficiency.

Various studies of the current detection mechanisms and their robustness to adversarial attacks provide valuable insights for this research. For instance, a study [1] discusses that GPT-2 detection could be augmented by integrating different statistical features, although it notes that further research is necessary. Another article [26] introduces a flexible method, Robust Density Estimation (RDE), for identifying adversarial examples in GPT-2 samples.

One of the novel detection methods is watermarking: embedding specific statistical patterns into the texts, which are invisible to the human eye but can be quickly picked by a detector [9]. Though watermarked text can quickly degrade under an adversarial attack, the method offers computational simplicity and might limit prompt engineering and parameter tweaking attack strategies. In contrast, more common attack approaches based on mutations [8] and paraphrasing [21] are less affected.

A promising improvement to the detector algorithms is the Zero-shot learning technique called DetectGPT, based on the observation that LLM-generated texts tend to occupy negative curvature regions of the model's log probability function [13]. Nevertheless, recent research [21] proves that the current state-of-the-art methods, such as watermarking [9], zero-shot detection [13] and detector neural networks, are not robust to paraphrasing and might not be robust to other adversarial techniques for GPT-2 generated texts.

Finally, a recent study on detector biases [11] shows that GPT-3.5 detectors are biased towards non-native speakers, as they tend to classify less diverse texts as AI-written. Additionally, the study shows that the accuracy of most state-of-the-art GPT-3.5 detectors can be decreased with a simple prompt asking the model to rewrite the text using literary language. These findings lay the groundwork for this research project.

Nevertheless, there is a noticeable gap in the literature: the majority of the aforementioned studies target GPT-2, and none consider the recently released Turnitin classifier, which is advertised as one of the most accurate and is widely used in the educational environment.

4 METHODOLOGY

The research is a mixed-methods study involving multiple experiments and manual text assessments. The experiments were designed to investigate the robustness of GPT-3.5 generated content detectors - Turnitin AI writing detector, OpenAI classifier and GPT-2 output detector against limited resource attack strategies. Due to the variety of experiments conducted, python code was written with the assistance of GPT-4 combined with extensive proofreading and testing. Additionally, some functionality had to be implemented manually due to GPT-4 limitations. The code and data are available on GitHub [4].

4.1 Research design

In order to answer the research question, first, accessible and resource-efficient attack strategies were selected: parameter tweaking, prompt engineering and character-level mutations. The selection is justified by resource efficiency and the high practicality of the above attacks, as anyone can reproduce them quickly. In contrast, most approaches presented in the literature are computationally heavy and require complex implementations, hindering their practicality.

The research involved generating text samples using different prompts and essay topics with the GPT-3.5 model and assessing the detection accuracy of the selected detectors. The dependent variable was the prediction score for the text being AI-generated, where 0% represents that the detector does not classify the text as AI-generated, and a 100% score would mean that the detector considers the text to be completely AI written. Each experiment was designed to test either a specific attack strategy or the modification of that strategy. The research process was exploratory, meaning that some of the experiments were sequential, and this paper presents them in the same sequence they were explored.

4.2 Experimental setup

The experiments are divided into three categories: parameter tweaking, prompt engineering, and character-level mutations. Parameter tweaking consists of experiments exploring how different GPT-3.5 model parameters affect the detection. Prompt engineering is based on qualitative text analysis of anomaly cases to derive a pattern that could be reliably replicated with a prompt. The character mutation category explores the resilience of the detectors towards traditional resource-efficient adversarial attack methods.

The research used the GPT-3.5-turbo model via OpenAI API for generating text samples. OpenAI regards the model as having improved performance and lower cost than other models from the GPT-3.5 family [17]. Additionally, for one of the experiments, texts were generated manually using GPT-4 because, as of June 2023, the GPT-4 API is in a limited beta. The text samples were produced as 500-word essays, with the topic taken from a list of 200 best essay topics [14]. The essay format was consistent across all experiments, though the precise prompts, essay genre, number of texts generated and corresponding topics varied between experiments. As the GPT-3.5 model requires specifying the number of tokens to be allocated for the request, the standard value for all of the experiments was set to 800 tokens, utilising the estimation of OpenAI that each token is roughly equal to 0.75 of a word [18]. Then, for the robustness of the text generation, the number of tokens was increased by 20%. However, the specific length of the text was checked in each experiment.

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

The text was regenerated if the number of words did not fall under the range specified separately in each experiment.

Most experiments followed the same prompt format to decrease environmental uncertainty: "Write a five-hundred-word argumentative essay on the topic 'topic'.", where the topic was selected from the list of essay topics defined above. Further in the text, this prompt format is referred to as a standard or baseline.

4.3 Measurement and Data collection

Three detectors were selected for the study: the Turnitin AI detector, the OpenAI classifier, and the GPT-2 Output detector. As the OpenAI classifier returns one of the five labels instead of a score, the results were mapped onto the numerical scales using the middle points of the corresponding score ranges provided by OpenAI [15]: "very unlikely" = 0.05, "unlikely" = 0.275, "unclear if it is" = 0.675, "possibly" = 0.94, "likely" = 0.99.

The Turnitin detector returns a score between 0 and 100, representing the confidence for the text being AI-generated. Consequently, the scores for Turnitin were kept without conversion. In contrast, the GPT-2 Output detector and OpenAI classifier (after mapping) scores were floats; thus, when the detectors were compared, the scores were multiplied by a hundred and rounded to the nearest digit.

Texts were uploaded to Turnitin and OpenAI detectors using Selenium in Python since, as of June 2023, the APIs to these detectors are not publicly available. The results were scraped similarly. Contrary, the GPT-2 Output detector was analysed using a Demo server, as the model is publicly available.

4.4 Data analysis

Due to diverse and skewed distributions of the results, a more qualitative analysis approach was performed instead of statistical tests, as the assumptions for most statistical tests were not satisfied. The procedure included generating boxplots, histograms and distribution diagrams using Python libraries: Matplotlib and Seaborn. The statistical significance of the results was not explored due to the aforementioned diversity and skewness of the data.

4.5 Parameter tweaking

This category of experiments explored the relation between GPT-3.5 model parameters and the detection rates of the three detectors. Firstly, the linear search over the parameter range was performed. Sequentially, the grid search was conducted based on the linear search results.

4.5.1 Linear search. Four parameters were selected from the list of parameters provided by the OpenAI [16]: Temperature, Top P, Frequency penalty, and Presence penalty. Temperature and Top P control randomness in the text, though the methods and range of acceptable values differ. The temperature ranges between 0.0 and 2.0 and defaults to 1.0 [16]. With an increment in value, the output becomes more random. However, it was found that increasing the value beyond the default of 1.0 made the model too unpredictable to test reliably. The length of the outputs started fluctuating beyond 20% of the tokens, and the quality of the texts was found to drop gradually. Consequently, the temperature was limited to the range of 0.0 - 1.0. Top P ranges from 0.0 to 1.0, defaults to 1.0, and represents the percentage of tokens selected based on their probability mass [16]. Generally, OpenAI recommends tweaking either temperature or top p since tweaking both makes the model unpredictable [16]. The frequency penalty ranges from -2.0 to 2.0, defaults to 0.0, and controls the frequency of tokens appearing in the text, with higher values leading to more diverse verbatim [16]. During the testing phase, it was found that increasing the frequency penalty beyond 1 degrades the quality of the texts rapidly. Additionally, as decreasing value beyond 0 increases the repetitiveness, the experiment range was set to 0.0 - 1.0. Finally, the presence penalty ranges from -2.0 to 2.0, defaults to 0.0, and controls the model's likelihood of repeating tokens in the text [16]. Higher values of presence penalty lead to the model producing more diverse texts and talking about new topics [16]. Following consideration similar to the frequency penalty, the negative values were discarded, and the range was set to 0.0 - 2.0.

The justification for selecting the four parameters lies in resource efficiency and the expected impact on the target text. Tweaking them is a simple procedure of setting a variable to a different value with substantial results. OpenAI provides other parameters, such as logit bias [16], but they are not considered in this research due to complexity.

Three topics from the argumentative essay genre were selected for text generation following the standard prompt format. A linear search was performed separately for each parameter with a step of 0.1 while fixating the other parameters. Three texts were generated for each step in the parameter range (one per essay topic), and their detection scores were recorded. The acceptable deviation of the text size was set to 10% of the original word count, and the sample was regenerated otherwise. Sequentially, the linear search for each parameter was plotted on a two-dimensional graph displaying the parameter value and average score of the three texts for that value. Three data sets were plotted for each value, corresponding to the three detectors. This would allow observing the parameter changes' effect on the detection and comparing the detectors' performance against prompt tweaking. A qualitative analysis was performed on the texts to identify the changes prompt tweaking induces on the quality of the texts.

4.5.2 Grid search. As frequency and presence penalties decreased detection rates in the first experiment, an extensive grid search was performed on their combinations within the ranges not impairing text quality: frequency penalty between 0.0 and 0.6; a presence penalty between 0.0 and 0.5. A grid search was performed with a step of 0.1; however, only a single text was generated for each parameter combination following the standard prompt for a single topic from the argumentative essay genre. The range for acceptable text length deviation was increased to 15% due to a larger text size fluctuation observed when tweaking both parameters. The results were plotted on a scattered plot with two parameters as axes and colour mapping indicating the score. Plots for each of the detectors were then combined to compare the performance.

4.6 Prompt engineering

The second branch of experiments explored the effect of GPT-3.5 and GPT-4 prompt engineering on detection rates. Firstly, a detection

comparison test was conducted between different essay genres. Secondly, the first experiment's results were used to design a prompt asking the model to imitate the discrepancy between the scoring of various texts and essay genres. Followingly, the effect of the new prompt on detection rates was explored. Intriguing quantitative and qualitative results of the experiment led to a test of the same prompt on the GPT-4 model, bringing forward new insights. A broader experiment on the GPT-3.5 model was designed sequentially based on the insights from the following article [11]. All of the experiments from this category utilised a single topic from the argumentative essay genre.

4.6.1 Genre difference. Two hundred essay topics were selected from the list of the best essay topics [14] and grouped by genres: argumentative, cause-effect, compare-contrast, controversial-argumentative, descriptive, expository, funny-argumentative, narrative, persuasive, and research. The standard prompt format was used for each topic, and the acceptable range of length deviation was set to 10% due to a smaller variation in sizes observed. The results were plotted as boxplots, violin plots and histograms, allowing for in-depth analysis.

Qualitative analysis of the differences between texts with lower and higher detection rates revealed a series of patterns distinguishing anomalous samples:

- Variability in sentence lengths and paragraph structures.
- Use of rhetorical questions and less formal language.
- Inclusion of contextual and personalization data.

4.6.2 Simple prompt engineering - GPT-3.5. Based on the qualitative analysis of the results of the genre difference test and anomalous samples with low detection rates, a prompt asking to replicate the patterns was devised and tested on GPT-3.5.

In this experiment, the essay formality requirement was omitted for both the baseline and modified prompts due to the informality pattern. Additionally, the acceptable length deviation was set to 15% due to larger fluctuations observed in the texts generated with the modified prompt.

Twenty texts were generated for both the standard prompt and the modified one using a single essay topic. The results were analysed qualitatively with box- and distribution plots for each detector. Overall, no difference was found between the detection rates.

4.6.3 Simple prompt engineering - GPT-4. The unexpected findings of the previous experiment resulted in a smaller-scale test of the modified prompt on GPT-4 to identify whether the results were due to the prompt being ineffective or the model not being able to follow it to the required extent. Due to the limitation of manual uploads, only ten texts were generated for each prompt, and the limit on length deviation was lifted. The results were analysed similarly to the GPT-3.5 test; however, only means are presented due to paper size limitations.

4.6.4 Advanced prompt engineering. A hypothesis emerged during the previous tests: providing regeneration instructions as a separate prompt might change the probability distribution of the worlds in the texts, which could lower the detection. This hypothesis was tested with the advanced prompt engineering experiment. Firstly, four prompts were defined:

- **Standard**
- **Standard second-query**: *"Regenerate the essay."*
- **Modified**: Refined prompt from previous experiments
- **Modified second-query**: *"Regenerate the essay. (Modified prompt insertion).*

Four essay groups were generated using the above prompts, containing ten texts with a text length deviation of 25% due to large fluctuations observed. Two groups were generated using a single query to the model. The other two groups utilised a two-query system, with the first query being the standard prompt and the second asking to rewrite following specific instructions. Additionally, the modified prompt for this test was the improved version of the previously used modified prompt, following the qualitative analysis of the texts produced during previous experiments.

4.6.5 Advanced prompt engineering - perplexity-burstiness. This experiment aimed to test a different approach to prompt engineering based on the commonly advertised prompt on the internet asking the model for increment in text perplexity and burstiness. Ten prompts were designed, and for each, ten texts were generated with a text length deviation of 25%. Prompts sent to the model ask to increase the burstiness and perplexity of the text in different ways. The first five prompts use single-query architecture, while the others utilise a two-query concept. The following methods were tested with different prompts:

- Explaining burstiness and perplexity, then asking to implement them.
- Explicitly asking to maximise either perplexity, burstiness or both.
- Explicitly asking to rewrite to avoid detection.

The last prompt followed the baseline format for comparison purposes. The results were analysed using boxplots, distribution plots, and mean histograms.

4.7 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 using ten argumentative essay topics with a text length deviation of 10%. Then, each mutation was applied to the samples, and the results were displayed visually.

5 RESULTS

5.1 Parameter tweaking

Parameter tweaking experiments aimed at exploring the relationship between parameter values and detection rates.

5.1.1 Linear search. The linear search experiment assessed how each variable independently influences detection rates and text quality.

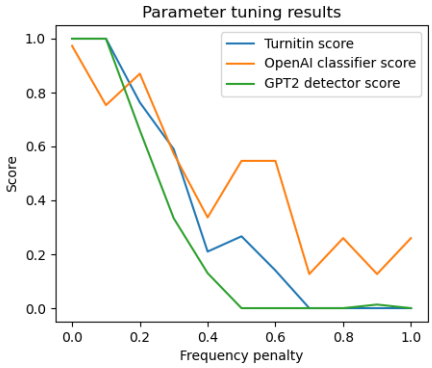

Fig. 1. Frequency penalty tuning.

Fig. 2. Presence penalty tuning.

Quantitatively, the detection rate of all three detectors dropped with an increment in either frequency or presence penalties (See Figures 1, 2). 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%, though some fluctuations were present. Expectedly, lowering the temperature and top p below the default resulted in more deterministic outputs leading to higher detection rates, while the increment resulted in model unpredictability.

Qualitatively, the increment in either the frequency or presence penalty led to the diversity of the texts spiking. A higher frequency penalty caused a wider vocabulary variety. Starting from a frequency penalty of 0.6-0.7, the occurrence of punctuation mistakes and unclear wordings skyrocketed. At the same time, the values below that range were found to increase the complexity of the texts while preserving quality and readability. In contrast, the presence penalty primarily influenced the diversity of perspectives and text engagement. However, starting from the range of 0.6-1.0, the coherence and logical progression plummeted, with texts becoming less focused and more exploratory.

To summarise, it was found that the increment of frequency penalty in the range of 0.0-0.6 and the presence penalty in 0.0-0.5 decreases detection rates while maintaining high text quality. A strong correlation between model parameters and detection rates was observed, with the detection rates plummeting following an increment in either one of the two parameters.

5.1.2 Grid search. The experiment aimed at researching combinations of frequency and presence penalties that could drop the detection rates while maintaining text quality. Figure 3 shows that the detection rates dropped for all three detectors with an increment in frequency and presence penalty. GPT-2 showed the worst performance among the three, with detection dropping to 0 even with a small parameter increment. Overall, the detection rates were found to drop with an increment in parameters, reaching an extremum at the upper portion of the parameter ranges.

5.2 Prompt engineering

This category of experiments explored the effectiveness of prompt engineering in bypassing detection.

5.2.1 Genre difference. The experiment tested whether there is a significant difference between detection in different essay genres. The texts with low detection rates were compared to the analogues with high detection, and a list of patterns that could contribute to lower detection was derived.

Statistically, no significant results could be derived as all distributions differed. However, the boxplot of the funny argumentative genre had a lower median across all three detectors (see Figures 4, 5, 6). Furthermore, the Interquartile ranges (IQR) for most styles differed across the detectors, with Turnitin having the smallest and GPT-2 - the largest.

Qualitative analysis of differences in the low and high detection samples resulted in the identification of the following patterns:

- Texts with lower detection had higher diversity in sentence lengths and paragraph structure.
- Texts with lower detection were found to employ rhetorical questions and less formal language more often.
- Texts with lower detection often included contextual and personalized elements.

5.2.2 Simple prompt engineering - GPT-3.5. Based on the results of the previous experiment, a prompt embodying identified patterns was designed and tested against a baseline. No significant differences were found in the detection rates, although the derived prompt showed a slightly higher average detection rate for all three detectors.

5.2.3 Simple prompt engineering - GPT-4. As simple prompt engineering on the GPT-3.5 model did not show any difference in detection rates, a similar test was conducted on GPT-4 model to test whether the prompt was ineffective or the model could not follow it to the needed extent. The texts crafted with the modified prompt resulted in a lower detection rate, with the mean falling close to zero for all three detectors (See Table 1). Additionally, it was found that the detectors struggle with text generated by the GPT-4 model, where OpenAI showed a mean probability near 30%, and the GPT-2 Output detector identified all the texts as overwhelmingly human-written.

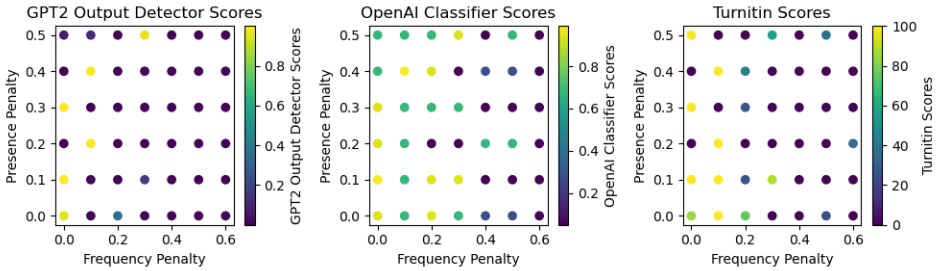

Fig. 3. Grid search results.

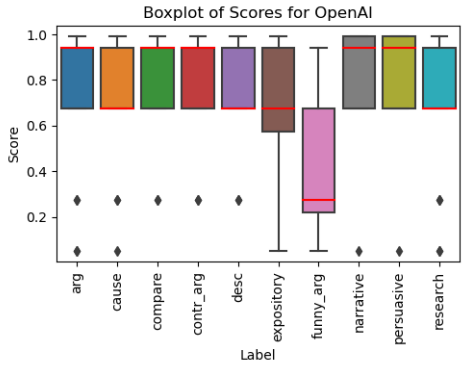

Fig. 4. Genre differences for OpenAI classifer

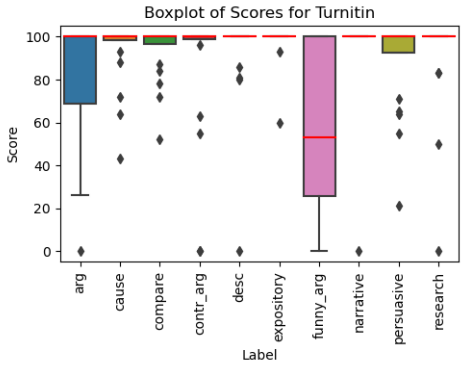

Fig. 6. Genre differences for Turnitin AI detector

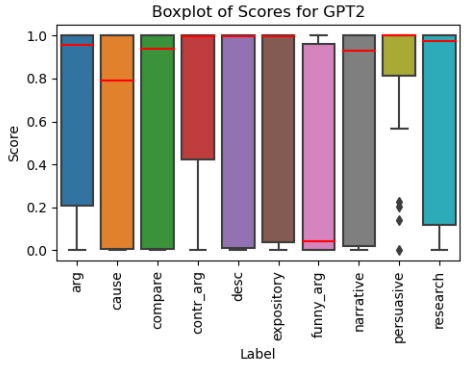

Fig. 5. Genre differences for GPT-2 Output detector

Table 1. GPT-4 prompt engineering mean detection scores (0-100)

	Openai	GPT-2	Turnitin
Normal	29	0	62
Smart	5	0	7

5.2.4 Advanced prompt engineering. Firstly, a hypothesis that the multi-query approach has lower detection rates than the single-query was tested. However, the regeneration experiment did not provide interesting insights, as no pattern could be derived from the distributions.

Secondly, a perplexity-burstiness approach was tested both with single-query and two-query implementation. The detection rate dropped for the texts produced utilizing a two-query approach and perplexity-burstiness prompts across the three detectors. The attack was found especially effective for the GPT-2 Output detector (See Figures 7, 8, 9), leading to a significant drop for all second-prompt attack methods in the experiment. Contrary, the OpenAI and Turnitin detectors performed better, with the median of the detections staying above 60% probability for the majority of attack methods. However, Turnitin had anomalous detection results for the second-prompt attack asking to improve both burstiness and perplexity without specifying the definitions. The median of the results lies at zero, with the upper quartile around 50% probability. Additionally, for the OpenAI, a label corresponding to the median values of second-prompt perplexity, burstiness, and both attacks is "unclear if it is", showing that such samples were not identified as AI-generated in most cases.

Overall, second-prompt perplexity-burstiness attack methods led to a decrement in detection rates across all detectors. The GPT-2 Output detector was impacted most, with the detection medians dropping under 20% probability. Additionally, the second-prompt burstiness-perplexity attack led to a median of 0% detection by the Turnitin detector.

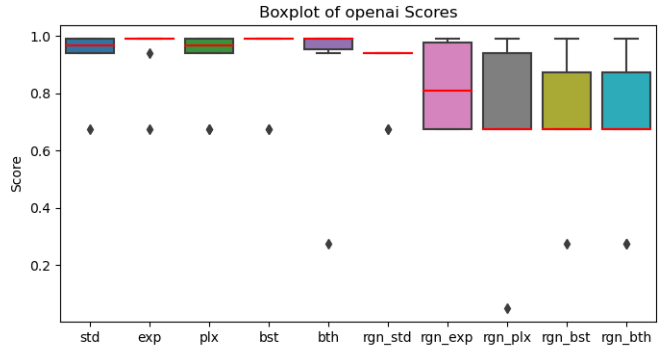

Fig. 7. GPT-3.5 prompt engineering effect on OpenAI classifier

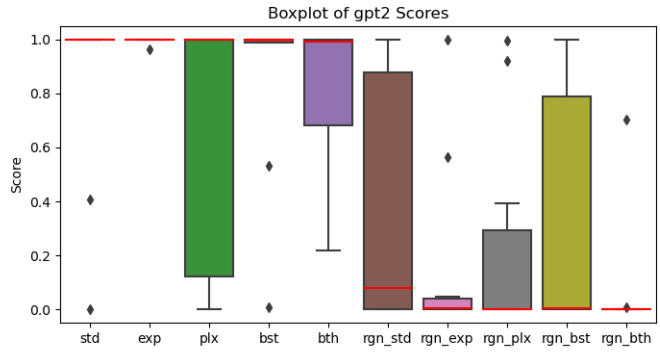

Fig. 8. GPT-3.5 prompt engineering effect on GPT-2 output detector

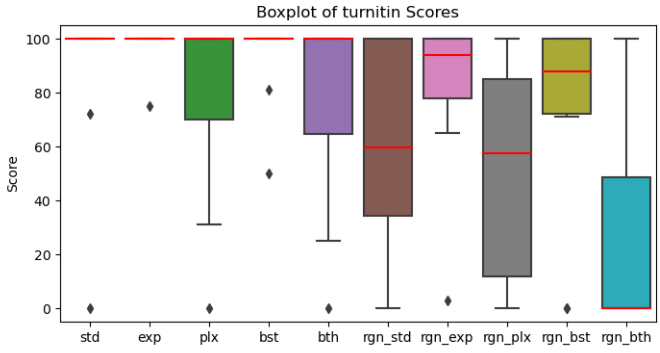

Fig. 9. GPT-3.5 prompt engineering effect on Turnitin AI detector

5.3 Character-level mutations

This experiment assessed the effectiveness of character-level mutations on the detection rate. Three mutations were conducted: replacing the Latin “a” with Cyrillic “a”, replacing the Latin “e” with Cyrillic “e”, and replacing the Latin lowercase “L” with uppercase “i”.

Table 2. Character-level mutation mean detection score (0-100)

	GPT-2	OpenAI	Turnitin
Standard	67	77	75.4
Replace a latin-cyrilic	0	52	Flag
Replace e latin-cyrilic	0	48	Flag
Replace l - i(uppercase) latin	0	38	2.1

All three mutations were found to be human-unnoticeable and dropped the detection rates significantly for the GPT-2 Output detector and OpenAI classifier (See Table 2). In contrast, Turnitin marked the first two mutations as a red flag and a potential fraud attempt, refusing to perform the detection. However, the third mutation was not flagged and effectively dropped Turnitin detection rates to 0 (See Table 2). Numerically, OpenAI performed better against character-level mutation attacks than the other two detectors (See Table 2). However, the textual labels corresponding to mean values range from “unlikely” to “unclear if it is”.

6 LIMITATIONS

Due to the nature of the data and limited resources, various limitations were identified, presenting future-research opportunities.

6.1 Timeframe

The experiments were conducted in May-June 2023. Thus, techniques found might become ineffective, leading to the irreproducibility of the results.

6.2 Sample size

The main limitation of the methodology is the sample size, where the number of texts generated per experiment ranged from 10 to 20 based on the experiment setup due to simulated manual uploads and web scraping. Nevertheless, the mixed-methods approach, careful design and consideration of limitations ensure that the findings offer valuable insights into the research topic.

6.3 Limited topics and genres

Most experiments except the essay genre difference test utilised 1-3 topics from the argumentative essay genre, which impacts the applicability of the results in the real-world environment.

6.4 Analysis

The samples’ distributions and variances were vastly different for each sample. Additionally, the distributions were skewed significantly and differently due to the specific nature of the data. Combined with the small sample size, this led to the inapplicability of statistical tests. Although the qualitative analysis of the graphs, means, and medians shows that some techniques are effective on the samples, the significance of the findings for the entire population could not be determined, limiting the real-world applicability. In future research, the bootstrapping technique could be tested to identify the statistical power of the results.

Secondly, the quality of the texts was assessed only by the author, leading to a possibility of a bias. Ideally, evaluation should be performed with multiple people involved. However, the task is challenging due to the large number of total texts produced. Additionally, automation procedures could be designed to partially overcome this limitation, which is a point for future research.

6.5 Regeneration and text length

Although all experiments used a prompt asking for a five-hundred-word essay, the deviation of text sizes fluctuated greatly, especially in parameter tweaking and regeneration experiments. Consequently, the allowed fluctuation of text sizes differed per experiment from 10% to 25% due to the observed discrepancies in text lengths for different parameters and prompts. The texts were regenerated if they did not fall under the specified range of length deviation. Grid search experienced a larger number of text regenerations. Thus, the deviation was set to 15%. The second-query approach from advanced prompt engineering tests produced the most regenerations, with some prompts requiring 10-15 regeneration attempts.

6.6 Parameter tweaking

Four parameters were selected as a base, and two were discarded due to the heavy impact on text quality and length. Thus, the solution proposed could be sub-optimal. Additionally, the parameter ranges selected and tested in the controlled environment are based on the observed influence of outside-range values on the text lengths and quality, which is limited to the sample and is not necessarily representative of reality. The linear search produced only three texts per set of parameter values, undermining real-world applicability. Furthermore, the grid search experiment utilised only a single text per parameter combination, preventing drawing conclusions separately. Finally, the step used for both searches is 0.1, which might not allow an accurate evaluation of the detection function.

Nevertheless, a similar strong trend was observed in both experiments, providing valuable insights into detection evasion.

6.7 Prompt engineering

Genre difference test explored ten common genres from essay format. However, the list does not cover the entire essay domain, and there might be genres with lower detection rates. Additionally, the prompt derived from the patterns observed in the experiment might not influence the GPT-3.5 model to replicate patterns consistently and to the needed extent.

7 CONCLUSION

This study explored the effectiveness of different practical adversarial attacks on the three AI-text detectors: Turnitin, OpenAI classifier, and GPT-2 Output detector. Based on the practicality assessment, three attack methods categories were selected: parameter tweaking, prompt engineering and character-level mutations. The results provided valuable insights into how different practical techniques could effectively manipulate the detection rates for GPT-3.5 texts.

Firstly, the relationship between frequency, presence penalties, and detection rates was apparent. An increment in any of the two parameters led to a decrease in detection rates at the cost of text coherence and quality. These findings implicate a tradeoff between text quality and detection rates. However, limiting the ranges of parameter tweaking and combining the two parameters revealed that detection can be reliably bypassed while still maintaining text quality.

Secondly, the study revealed how specific patterns identified from low-detection texts could be exploited to improve detection evasion. These include a higher diversity in sentence length and paragraph structure, utilisation of rhetorical questions, informal language, and addition of personalised and contextual elements. However, prompt engineering aimed at replicating these patterns had mixed results, showing little to no improvement for GPT-3.5 and significant improvement for GPT-4 model. Additionally, it was found that the researched detectors struggle even with standard GPT-4 outputs, with Turnitin having a mean under 60% probability and the GPT-2 Output detector not being able to identify a single text piece.

Thirdly, the study explored burstiness-perplexity prompt attacks with single- and double-query approaches. The findings revealed that applying a second-query approach consistently decreased detection rates, especially for the GPT-2 Output detector, where the detection medians dropped to zero. In contrast, Turnitin and OpenAI displayed higher robustness towards second-query methods, with medians of probability not dropping under 60% for most samples. However, in the case of OpenAI, the corresponding textual label median was equal to “unclear if it is AI-written”, undermining the detection performance. The phenomenon of second-query attack effectiveness could be explained by a larger input to the GPT-3.5 model, including the first text generated. However, this hypothesis requires further testing. A detection anomaly was also observed in Turnitin, where the second-query attack asking to rewrite the text while increasing both perplexity and burstiness resulted in a median detection rate of zero. This questions the reliability of Turnitin against the specific attack and provides an opportunity for future research to identify the potential causes.

Fourthly, the study revealed the effectiveness of character-level mutation attacks for detection evasion. All three mutations dropped detection rates for OpenAI and GPT-2 Output detectors, while Turnitin managed to identify two of the three attacks. Nevertheless, replacing the lowercase “L” with a capital “i” evaded detection and plummeted detection rates. However, as these mutations are simple and easy to implement, detection algorithms are expected to adapt to identify and flag these types of mutations in the future.

Several limitations were identified in the study, including small sample sizes, topic-specific focus, lack of qualitative evaluators, and inability to explore the statistical significance of the results. The above limitations could hinder the applicability of findings to real-world situations, and the results might not apply to other models, detectors and text styles. Future research is required to confirm the findings while addressing the limitations.

In conclusion, this study offers valuable insights into the manipulation of AI text generation and text mutations for detection evasion. The need to improve existing detectors to keep pace with the evolution of generation models is highlighted. The research findings contribute to the broader discussion on AI detection, revealing vulnerabilities of existing detectors.

ACKNOWLEDGMENTS

The author would like to thank Daniel Braun for supervising this project and providing useful insights.

REFERENCES

[1] 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 (3 2022). https://doi.org/10.1109/IJCNN55064.2022.9892269

[2] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2018. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. *Proceedings of the 28th USENIX Security Symposium* (9 2018), 321–338. https://arxiv-org.ezproxy2.utwente.nl/abs/1809.02861v4

[3] Yinpeng Dong, Shuyu Cheng, Tianyu Pang, Hang Su, and Jun Zhu. 2022. Query-Efficient Black-box Adversarial Attacks Guided by a Transfer-based Prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (3 2022), 9536–9548. Issue 12. https://doi.org/10.1109/TPAMI.2021.3126733 image
.

[4] Yali Du, Meng Fang, Jinfeng Yi, Jun Cheng, and Dacheng Tao. 2018. Towards Query Efficient Black-box Attacks: An Input-free Perspective. *Proceedings of the ACM Conference on Computer and Communications Security* 18 (9 2018), 13–24. https://doi.org/10.1145/3270101.3270106

[5] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. HotFlip: White-Box Adversarial Examples for Text Classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* 2 (12 2017), 31–36. https://doi.org/10.18653/v1/p18-2006

[6] 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* (8 2018), 50–56. https://doi.org/10.1109/SPW.2018.00016

[7] 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* (12 2014). https://arxiv-org.ezproxy2.utwente.nl/abs/1412.6572v3

[8] 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* (7 2019), 8018–8025. https://doi.org/10.1609/aaai.v34i05.6311 text: text fooler. text problem: discrete nature
.

[9] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A Watermark for Large Language Models. (1 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2301.10226v3

[10] Gongbo Liang, Jesus Guerrero, and Izzat Alsmadi. 2023. Mutation-Based Adversarial Attacks on Neural Text Detectors. (2 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2302.05794v1

[11] Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023. GPT detectors are biased against non-native English writers. (4 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2304.02819v2

[12] Nicholas A. Lord, Romain Mueller, and Luca Bertinetto. 2022. Attacking deep networks with surrogate-based adversarial black-box methods is easy. (3 2022). https://arxiv-org.ezproxy2.utwente.nl/abs/2203.08725v1

[13] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. (1 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2301.11305v1

[14] A Nova. 2019. 100+ Essay Topics for an Outstanding Essay (2022). https://www.5staressays.com/blog/essay-writing-guide/essay-topics

[15] OpenAI. 2023. AI Text Classifier - OpenAI API. https://platform.openai.com/ai-text-classifier

[16] OpenAI. 2023. API Reference - OpenAI API. https://platform.openai.com/docs/api-reference/chat/create

[17] OpenAI. 2023. Models - OpenAI API. https://platform.openai.com/docs/models/gpt-3-5

[18] OpenAI. 2023. Pricing. https://openai.com/pricing

[19] 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 (5 2023), 101539. Issue 5. https://doi.org/10.1016/J.JKSUCI.2023.03.017

[20] 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* (1 2021). https://doi.org/10.1109/IJCNN48605.2020.9207272

[21] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can AI-Generated Text be Reliably Detected? (3 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2303.11156v1

[22] 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. (2023).

[23] 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* (12 2013). https://arxiv-org.ezproxy2.utwente.nl/abs/1312.6199v4

[24] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. (2 2023). https://arxiv-org.ezproxy2.utwente.nl/abs/2302.11382v1

[25] Mohammad Mehdi Yadollahi, Arash Habibi Lashkari, and Ali A. Ghorbani. 2021. Towards Query-efficient Black-box Adversarial Attack on Text Classification Models. *2021 18th International Conference on Privacy, Security and Trust, PST 2021* (2021). https://doi.org/10.1109/PST52912.2021.9647846

[26] 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* (3 2022), 3656–3672. https://doi.org/10.18653/v1/2022.findings-acl.289

[27] Zheng Yuan, Jie Zhang, and Shiguang Shan. 2021. Adaptive Image Transformations for Transfer-based Adversarial Attack. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13665 LNCS (11 2021), 1–17. https://doi.org/10.1007/978-3-031-20065-6_1 image
.