SilverSpeak: Evading AI-Generated Text Detectors using Homoglyphs

Aldan Creo
Dublin, Ireland
aldan.creo@rai.usc.es

Shushanta Pudasaini
Institute of Engineering (IOE)
Kathmandu, Nepal
shushanta574@gmail.com

Abstract

The advent of Large Language Models (LLMs) has enabled the generation of text that increasingly exhibits human-like characteristics. As the detection of such content is of significant importance, substantial research has been conducted with the objective of developing reliable AI-generated text detectors. These detectors have demonstrated promising results on test data, but recent research has revealed that they can be circumvented by employing different techniques. In this paper, we present homoglyph-based attacks ('A' → Cyrillic 'A') as a means of circumventing existing detectors. We conduct a comprehensive evaluation to assess the effectiveness of these attacks on seven detectors, including ArguGPT, Binoculars, DetectGPT, Fast-DetectGPT, Ghostbuster, OpenAI's detector, and watermarking techniques, on five different datasets. Our findings demonstrate that homoglyph-based attacks can effectively circumvent state-of-the-art detectors, leading them to classify all texts as either AI-generated or human-written (decreasing the average Matthews Correlation Coefficient from 0.64 to -0.01). Through further examination, we extract the technical justification underlying the success of the attacks, which varies across detectors. Finally, we discuss the implications of these findings and potential defenses against such attacks.

1 Introduction

LLMs have soared in popularity in a wide variety of domains as their text generation capabilities become increasingly human-like (Bin-Nashwan et al., 2023). For instance, it is estimated that the percentage of arXiv articles whose abstract has been revised by ChatGPT is around 35% (GENG and Trotta, 2024). While LLMs can prove beneficial (Ngo, 2023), there is growing concern about their potential misuse (Sullivan et al., 2023; Yan et al., 2024; Li et al., 2023; Sebastian, 2023).

Thus, a number of approaches to detect AI-generated text have been proposed, including zero-shot classifiers (Gehrmann et al., 2019; Mitchell et al., 2023; Bao et al., 2024; Hans et al., 2024; Su et al., 2023), binary classifiers (Solaiman et al., 2019; Verma et al., 2024; Liu et al., 2023), and watermarking techniques (Zhu et al., 2024; Giboulot and Teddy, 2024; Zhang and Koushanfar, 2024; Molenda et al., 2024; Wu et al., 2023).

At the same time, research has been conducted on methods for circumventing AI-generated text detectors. Some popular techniques include paraphrasing (Krishna et al., 2023; Peng et al., 2023), watermark stealing (Jovanović et al., 2024), Substitution-based In-Context example Optimization (Lu et al., 2024), reinforcement learning (Nicks et al., 2023) or space infiltration (Cai and Cui, 2023). In this paper, we study an alternative technique
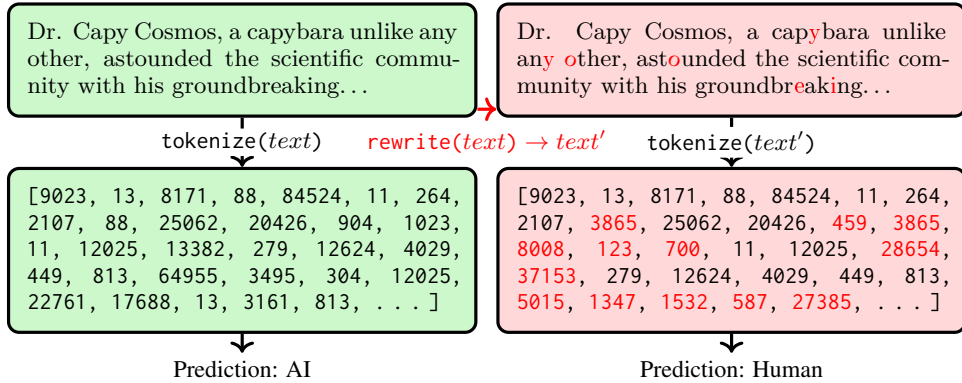
Figure 1: Homoglyph-based attack. The left box shows the original text, adapted from (Hans et al., 2024), and the right box shows the text after rewriting some of its characters. The bottom boxes show the tokenized versions from (OpenAI, 2024b). Differences are shown in red.

based on homoglyphs.

Homoglyphs are visually similar characters with different encodings (e.g., Latin 'a' and Cyrillic 'a') (Ginsberg and Yu, 2018). This allows us to generate **rewritten versions of any given text that can evade AI-generated text detectors** (Figure 1). Kirchenbauer et al. (2023) identified the usage of homoglyphs as a potential avenue for evading AI-generated text detectors. However, to the best of our knowledge, no study has yet conducted a comprehensive evaluation of the effectiveness of this approach across diverse datasets and detectors, nor has it provided insights into the technical justification of homoglyph-based attacks, a gap that we aim to fill in this paper.

Our **main contributions** are:

- **What** *are homoglyph-based attacks?* We introduce them as a way to evade AI-generated text detectors.

- **How much** *can homoglyph-based attacks affect AI-generated text detectors?* We evaluate their effectiveness on five datasets and seven detectors in Section 2. In Section 3, we show that they can bring average Matthews Correlation Coefficients from 0.64 to -0.01. This shows a complete eva-

sion, discussed in Section 4.1.

- **Why** *do homoglyph-based attacks work?* We analyze and justify such performance decline in Section 4.2.

- *What are the **ethical implications** of these findings?* We discuss them in Section 6, along with possible **defenses** against such attacks.

- Additionally, we introduce the first publicly available dataset of homoglyph-based attacks targeting AI-generated text detectors.

2 Methods

In this section, we delineate our experimental approach, along with a description of the detectors and datasets employed. We make our code and datasets publicly available at https://github.com/ACMCMC/silverspeak, under CC BY-SA 4.0 and ODC-BY licenses. Furthermore, we ensured that our study adheres to the intended usages of the detectors and datasets presented, for which we include licensing information below.

2.1 Experiments

As shown in Figure 2, we evaluate the effectiveness of homoglyph-based attacks
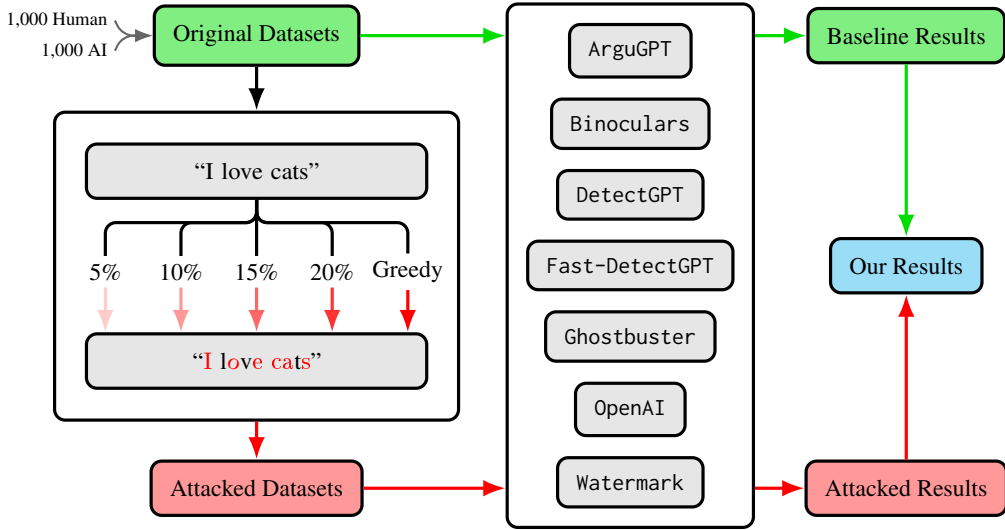
Figure 2: Our experimental process. First, we generate a set of **rewritten datasets** by applying homoglyph-based attacks, with varying replacement percentages, on all original datasets. Then, we run the **detectors** on the original and attacked datasets to get the metrics presented.

on seven detectors and five datasets, each with 2,000 samples (1,000 human and 1,000 AI). We used the original text and five attacked versions, generated by replacing **5%**, **10%**, **15%**, and **20%** of randomly chosen characters in the text (random attack), or all of the possible characters that can be replaced (**greedy** attack).

We also conducted initial experiments on an optimized setting where we perform replacement only on tokens that have the highest loglikelihoods (those that are most likely to be AI-generated) when evaluated by an LLM. However, given that the previous attacks are already effective (Section 3), we decided to focus on them for the rest of the experiments, as they are less computationally expensive and do not vary depending on the choice of the LLM.

We utilized the homoglyphs provided in (Davis and Suignard, 2023). We based our code on the Hugging Face Transformers and Datasets libraries with PyTorch as backend (Wolf et al., 2019; Paszke et al., 2019). We executed the experiments on a NVIDIA A100, for which we present utilization by each detector in Table 1.

Detector	Time	Space
ArguGPT	2	5.2
Binoculars	6	34.2
DetectGPT	276	10.8
Fast-DetectGPT	25	19.5
Ghostbuster	240	0
OpenAI	2	5.2
Watermark	3	8.6

Table 1: Approximate requirements on time and space for one experiment, in minutes and gigabytes. We report on the unattacked *reuter* dataset as times do not vary significantly across datasets. For a full experiment suite on a detector, the time requirement is multiplied by the number of attacks and datasets.

2.2 Detectors

We conduct experiments on:

- ArguGPT: A RoBERTa-based classifier trained on a dataset of human and AI-generated arguments (Liu et al., 2023). We utilize the sentence-level model under a MIT license.

- Binoculars: Computes a ratio of the perplexity measured on an LLM

and its cross-entropy with the perplexity of another. The text is determined to be AI-generated or not by comparing the ratio with a chosen threshold. We utilize the code (BSD 3 license), default threshold (`low-fpr`), and models (observer `falcon-7b`; performer `falcon-7b-instruct`) provided by the authors (Hans et al., 2024).

- `DetectGPT`: Compares the likelihood of an input text with a series of AI-perturbed versions, assuming that loglikelihoods will drop more for AI-generated texts (Mitchell et al., 2023). We utilize the open-source implementation (MIT license) by (Tayyab, 2023), with `GPT-2-Medium` and `T5-Large` as scoring and rewriting models (Radford et al., 2019; Raffel et al., 2020).

- `Fast-DetectGPT`: An optimization that measures the conditioned probability of each token against its alternatives, rather than among texts. This means that only one forward pass is needed to score the perturbed tokens, rendering it much faster (Bao et al., 2024). It has been released under a MIT license.

- `Ghostbuster`: A classifier trained on a set of forward-selected features based on token probabilities measured on weak language models (Verma et al., 2024), licensed under CC BY 3.0. We use its web interface, as described in Section 7.

- `OpenAI`'s detector: A `RoBERTa`-based classifier trained on a large dataset of human and AI-generated texts (Solaiman et al., 2019). We utilize the large variant (MIT license).

- `Watermark`: Based on a *lefthash* algorithm, which computes a hash of the previous token and uses it to shift the next token logits, so that this skewed distribution can be detected (Kirchenbauer et al., 2023). We use the Hugging Face implementation (Apache 2.0) (Wolf et al., 2019).

2.3 Datasets

We derived our datasets as follows:

- *essay, writing prompts, reuter*: Derived from (Verma et al., 2024), also utilized by (Hans et al., 2024). The *essay* dataset consists of essays from IvyPanda. The *writing prompts* dataset consists of prompts from the subreddit r/WritingPrompts. The *reuter* dataset consists of news articles from the Reuters 50-50 authorship identification dataset. They are licensed under CC BY 3.0.

- *CHEAT*: Abstracts of academic papers, derived from (Yu et al., 2024) under a MIT license.

- *realnewslike*: Derived from the C4 realnewslike dataset (Raffel et al., 2020) (ODC-BY license). We generate 200-token watermarked completions with `OPT-1.3B` (Zhang et al., 2022), as in (Kirchenbauer et al., 2023), taking 8 minutes and 23.9 GB on a NVIDIA A100. The nature of this dataset is such that it is only used to test the `Watermark` detector, as others cannot detect watermarks.

To ensure that all datasets have the same number of examples, we randomly select 1,000 human and 1,000 AI-written examples from each source dataset. We do not split the datasets as our study does not require training any model.

3 Results

We summarize our experimental results in Table 2. Full results are reported in

Dataset	Detector	Original	5%	10%	15%	20%	Greedy
CHEAT	ArguGPT	0.94	0.0	0.0	0.0	0.0	0.0
	Binoculars	0.93	0.37	0.11	0.04	0.02	0.13
	DetectGPT	0.14	-0.02	0.03	0.13	0.06	0.0
	Fast-DetectGPT	0.9	0.23	0.04	0.02	0.0	-0.01
	Ghostbuster	0.64	0.41	0.32	0.12	0.06	0.02
	OpenAI	0.47	0.0	0.0	0.0	-0.02	0.0
essay	ArguGPT	0.92	0.0	0.0	0.0	0.0	0.0
	Binoculars	0.91	0.22	0.05	0.0	0.0	0.05
	DetectGPT	0.24	-0.01	0.11	0.21	0.08	0.0
	Fast-DetectGPT	0.88	0.22	0.04	0.0	0.0	-0.08
	Ghostbuster	0.92	0.73	0.51	0.13	0.0	0.0
	OpenAI	-0.21	0.0	0.0	0.0	0.0	0.03
reuter	ArguGPT	0.92	0.0	0.0	0.0	0.0	0.0
	Binoculars	0.8	0.22	0.07	0.03	0.02	0.08
	DetectGPT	0.23	0.0	0.03	0.34	0.14	0.0
	Fast-DetectGPT	0.92	0.28	0.1	0.02	0.0	0.04
	Ghostbuster	0.93	0.61	0.51	0.16	0.04	0.0
	OpenAI	0.27	0.0	-0.04	-0.09	-0.11	-0.06
writing prompts	ArguGPT	0.39	0.0	0.0	0.0	0.0	0.0
	Binoculars	0.85	0.2	0.0	0.0	0.0	-0.04
	DetectGPT	0.44	0.04	0.01	0.02	0.02	0.0
	Fast-DetectGPT	0.79	0.3	0.05	-0.03	0.0	-0.33
	Ghostbuster	0.88	0.42	0.64	0.33	0.09	0.0
	OpenAI	-0.05	-0.04	-0.05	-0.13	-0.11	0.01
realnewslike	Watermark	0.92	0.18	-0.01	0.0	-0.03	0.0
Average		0.64	0.17	0.1	0.05	0.01	-0.01
Standard deviation		0.36	0.21	0.19	0.11	0.05	0.08

Table 2: Matthews Correlation Coefficient (MCC) of all detectors on all datasets for all attack configurations. The color of the cell represents its value, clipped between 0 (red) and 1 (green).

Appendices A and B, with the raw results available in our released datasets.

The results correspond to a single run, as we confirmed that the scores obtained are identical across multiple executions.

It should be noted that some conventional metrics employed to assess the efficacy of detectors may prove to be deceptive in this particular setting. For example, Figure 13e shows a confusion matrix where the F1 score is 0.67, but the detector is classifying almost all examples as AI-generated. We argue that the Matthews Correlation Coefficient (MCC) is better suited (in this case, 0.08), placing greater emphasis on class balance (Baldi et al., 2000). Therefore, we use it as our main metric, and advise caution when interpreting the results based on other metrics in the appendices. MCC yields values from -1 (inverse correlation) to 1 (perfect corre-

lation), with 0 representing no correlation.

4 Discussion

In this section, we discuss the results obtained from the experiments conducted on different AI-generated text detectors using homoglyph-based attacks. Then, we analyze the effectiveness of the attacks and their technical justifications.

4.1 Effectiveness of the attacks

Baseline performance varies across detectors and datasets. Before the attacks, MCC values range from -0.21 to 0.94, with an average of 0.64 and a standard deviation of 0.36.

Binoculars and Fast-DetectGPT show consistently high MCCs. ArguGPT and Ghostbuster show a wider range of MCCs across datasets, with DetectGPT and the OpenAI detector having lower

baseline scores. The `Watermark` detector shows a high baseline MCC, albeit only tested on the *realnewslike* dataset.

We performed a side exploration on the low scores of `DetectGPT` and the `OpenAI` detector. We found that their scores can be improved by adapting their classification thresholds to each dataset they are applied on. However, including these results would mean deviating from the original implementations, so we abstained from changing the thresholds in our study.

Interestingly, not all detectors are affected in the same way by the attacks. Generally, we observe two distinct trends when applying the attacks:

- The detector tends to classify the examples as **human-written**, even when they are AI-generated. This happens on all replacement percentages, but even more prominently as the percentage of replacements increases. This is the case for `ArguGPT`, `Binoculars`, `Fast-DetectGPT`, the `OpenAI` detector, and `Watermark`.

- The detector classifies more examples as **human-written on low replacement percentages** (5% or 10%). However, as the percentage of replacements increases, the detector starts classifying the examples as AI-generated. On intermediate replacement percentages, the detector tends to behave as a random classifier. This temporarily increases the MCC, as some examples are classified correctly. Then, the tendency reaches a plateau and **higher percentages** (20%, greedy attack) cause the detector to classify almost all examples as **AI-generated**. This is the case for `DetectGPT`, which plateaus around 15%, and `Ghostbuster`, which plateaus around 10%.

While the behavior of the detectors varies, the effectiveness of the attacks is consistent across all detectors and datasets, showing a pronounced decline in performance. Lowest MCCs are observed in the greedy replacement setting, where **the attack consistently** (standard deviation of 0.08) **renders detectors ineffective** (average of -0.01).

4.2 Technical justification

In this section, we provide insights into the effectiveness of the attacks, separately exploring each group of detectors.

4.2.1 Perplexity-based models

`Binoculars`, `DetectGPT` and `Fast-DetectGPT` are based on perplexity, shown in Equation 1 (Alon and Kamfonas, 2023). Let N be the number of tokens in the text, and $p(t_i)$ the probability of token t_i given $t_1, \ldots, t_{i-1}$, according to an LLM.

$$
\text{Perplexity} = \exp\left[-\frac{1}{N}\sum_{i=1}^{N}\log p(t_i)\right]
\tag{1}
$$

As homoglyphs have different encodings, tokenizers treat them differently. Two observations can be made:

1. Since the training corpora used to train popular tokenizers (such as those based on Byte-Pair Encoding (Sennrich et al., 2016)) do not often contain sequences that mix characters from different alphabets, it is likely that attacked tokens will be split into smaller ones: N increases.

2. Since the attacked sequence of tokens does not resemble the training data, the loglikelihoods for attacked tokens will generally be lower.

Therefore, the summation contains **more tokens** ($\uparrow N$) **with lower loglikelihoods** ($\downarrow \log p(t_i)$), increasing perplexity.

(a) Differences in loglikelihood per token
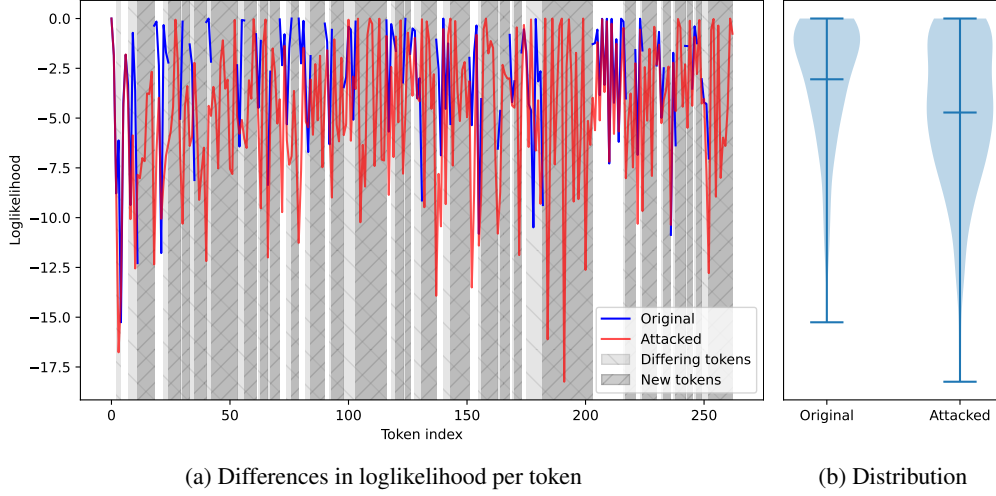
(b) Distribution

Figure 3: Token loglikelihoods for the text in Figure 1 on `BLOOM-560m` (Le Scao et al., 2023). The attacked text (10% replacement) has a distribution shifted towards more negative values.

Figure 3 illustrates the impact of homoglyph-based attacks on tokens and their associated log likelihoods. In this example, modifying **10%** of the characters in the text changes their tokenization **70%** of the time. The attacked text exhibits a more negative loglikelihood distribution than the original text, as shown in Figure 3b. Therefore, the attacked text appears "more likely to be human" when the perplexity is evaluated with an LLM, while keeping the same appearance. In summary, homoglyph-based attacks are effective at **shifting the distribution of loglikelihoods towards more negative values, which can evade detection**.

4.2.2 Classification models

`ArguGPT` and the `OpenAI` detector are `RoBERTa`-based models with a classification head (Liu et al., 2023; Solaiman et al., 2019). We hypothesize that the presence of homoglyphs in the text causes the output embeddings to become much less discriminative, as the model is unable to understand the semantics of the text.

To test this hypothesis, we remove the classification head from `ArguGPT` and ob-
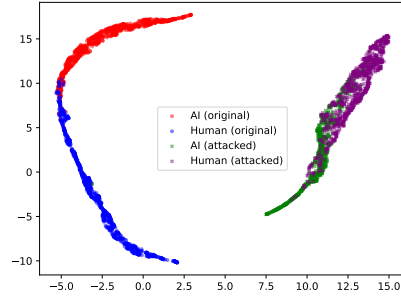
Figure 4: Embeddings from `ArguGPT`. While the original texts are well-separated, the embeddings of the attacked texts are mixed and placed in a different subspace.

tain the mean of the embeddings for the original and attacked texts (10% replacement) on the *CHEAT* dataset. We then reduce their dimensionality to 2D with UMAP (McInnes et al., 2018). We set the local connectivity to 5, minimum distance to 0.1, and number of neighbors to 15. We plot the embeddings in Figure 4.

Three clusters can be observed. Two of them correspond to the original texts, where AI and human texts are clearly separated. However, the third cluster (green

and purple) corresponds to the attacked texts, where embeddings are mixed. This indicates that the classification head is fed with discriminative embeddings in a normal scenario, but with homoglyphs, **the embeddings are less discriminative and placed in an unseen region of the space, therefore leading to misclassifications**.

As for Ghostbuster, a similar justification to Section 4.2.1 can be made. The model is based on a linear classifier, and while it does not use perplexity, the features it is trained on are based on the probability of generating each token in the text under several weaker language models (Verma et al., 2024). Therefore, the same principles apply: the presence of homoglyphs in the text alters the calculation of the probabilities, leading to a **shift in the distribution of the features used by the classifier that evades detection**.

4.2.3 Watermarking

The Watermark detector is a special case, as it is not designed to analyze the semantics of the text, or any of its features, other than the presence of a watermark. The probability of a text having been generated with a watermark is calculated with a one-proportion z-test, as shown in Equation 2. Given a text, $|s|_G$ is the number of *green* ("expected") tokens and T is the total number. γ is a hyperparameter, the probability of a token being green (Kirchenbauer et al., 2023).

$$z = (|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)} \quad (2)$$

If the text is generated with knowledge of the watermark rule, we can expect $|s|_G$ to deviate significantly from γT; *i.e.* the sample mean will be higher than the expected mean, leading to a high z value. This is the case for the original texts, where the watermark is easily detected.

However, results show that watermarks are highly sensitive to changes in the text.

This is due to the fact that the *lefthash* watermarking algorithm is based on a simple scheme where a list of green tokens is generated using the previous token $t-1$ (Kirchenbauer et al., 2024). This list is used to shift the logits of the current token t, so that the distribution is skewed towards green tokens. As homoglyph-based attacks alter tokenization, many of the green lists are generated with different seeds, and the probability of t being green becomes γ, as in a human-written text. Moreover, even if $t-1$ remains the same, if t is changed, its probability of being green is also γ. The two factors combined lead to a **significant decrease in the number of green tokens**, thus reducing z and rendering the watermark undetectable.

5 Conclusion

This paper demonstrated that homoglyph-based attacks can evade state-of-the-art AI-generated text detectors. We performed a systematic evaluation of the effectiveness of these attacks on seven different detectors and five different datasets. Our results showed that homoglyph-based attacks are very effective, to the point that their MCC drops to around 0 (no correlation) in all of them, albeit at different replacement percentages. We then analyzed the internal mechanisms of the detectors to provide a technical justification for the effectiveness of the attacks. Furthermore, we have publicly released our implementation and datasets, which we hope will facilitate further research on AI-generated text detection algorithms. The effectiveness of these attacks adds to the existing evidence that existing AI-generated text detectors are unfit for purpose, highlighting the need for more robust detection mechanisms. The proposed attacks can be employed to assess the resilience of future AI-generated text detectors and to develop more effective solutions.

6   Ethical impact and safeguards

Our work has significant ethical implications, including the potential for increased instances of academic misconduct, misinformation, and social engineering (Majumdar et al., 2024).

Furthermore, while alternative methods such as paraphrasing necessitate the use of LLMs (Krishna et al., 2023), homoglyph-based attacks can be conducted with a simple script and minimal computational resources. This **lower barrier for access** exacerbates their potential impact.

It is not our intention to encourage malicious usage; rather, we seek to contribute to the growing evidence on the unreliability of current detectors (Sadasivan et al., 2024; Yan et al., 2024) and promote the design of sturdier ones. It is **deeply concerning** that a number of commercially available tools like Undetectable (2024) are widely used in sectors like academia, yet they are vulnerable to an attack that can be executed with minimal effort.

Fortunately, it is possible to forestall these attacks by incorporating additional **safeguards** into the detection process. Input constraints, such as limiting the character set that can be utilized (Ginsberg and Yu, 2018) or mapping them to a standard form (Alvi et al., 2017), can be an effective mitigation strategy in several cases.

Other contexts may require more sophisticated solutions. For instance, scientific articles frequently contain Greek symbols in their discourse, which should not be treated as indicators of homoglyph-based attacks. Instead, one possibility is to analyze loglikelihood scores (Figure 3) (Alon and Kamfonas, 2023), while another is to consider architectures based on neural networks (Woodbridge et al., 2018) or optical character recognition (Ginsberg and Yu, 2018). No universal solution exists, and the choice should be based on the nature of the text and detector.

7   Limitations

Our work has some limitations that should be considered when interpreting results.

Optimized attacks   As our aim was to justify and assess the extent to which homoglyph-based attacks are able to evade AI-generated text detectors, we did not attempt to optimize (Section 2.1). It may be possible to achieve the same evasion rates with lower replacement percentages by strategically selecting the characters to replace. Additionally, there may be merit in studying character sets other than homoglyphs (Boucher et al., 2022).

Datasets   We are confident that the number of samples per dataset (2,000) is enough to demonstrate the effectiveness of the attacks, as the results do not elicit the need for further exploration (we observe a complete evasion of the detectors with a low standard deviation). However, generalizability to languages other than English remains to be studied, where homoglyphs may be naturally present. Nonetheless, if detectors tend to misclassify non-native English writing samples as AI-generated (Liang et al., 2023), we expect that homoglyph-based attacks would be effective in other languages as well.

`Ghostbuster` deprecation   Another limitation is that `Ghostbuster` is based on the deprecated `ada` and `davinci` models (OpenAI, 2024a). This prevents us from running it on our infrastructure, and while we have tried to contact the authors for a solution, we have not received a response yet. Surprisingly, the web interface provided by the authors remains operational, enabling us to evaluate the detector. However, we are unable to confirm the models currently in use, and therefore cannot guarantee that the results are consistent with those presented in the original paper, nor that they will remain reproducible.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *Preprint*, arXiv:2308.14132.

Faisal Alvi, Mark Stevenson, and Paul Clough. 2017. Plagiarism detection in texts obfuscated with homoglyphs. In *Advances in Information Retrieval*, pages 669–675, Cham. Springer International Publishing.

Pierre Baldi, Søren Brunak, Yves Chauvin, Claus A. F. Andersen, and Henrik Nielsen. 2000. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-DetectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

Saeed Awadh Bin-Nashwan, Mouad Sadallah, and Mohamed Bouteraa. 2023. Use of ChatGPT in academia: Academic integrity hangs in the balance. *Technology in Society*, 75:102370.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible NLP attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004.

Shuyang Cai and Wanyun Cui. 2023. Evade ChatGPT detectors via a single space. *arXiv preprint arXiv:2307.02599*.

Mark Davis and Michel Suignard. 2023. Unicode security mechanisms. Technical standard, Unicode.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander 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, Florence, Italy. Association for Computational Linguistics.

Mingmeng GENG and Roberto Trotta. 2024. Is chatGPT transforming academics' writing style? In *ICML 2024 Next Generation of AI Safety Workshop*.

Eva Giboulot and Furon Teddy. 2024. WaterMax: breaking the llm watermark detectability-robustness-quality trade-off. *arXiv preprint arXiv:2403.04808*.

Avi Ginsberg and Cui Yu. 2018. Rapid homoglyph prediction and detection. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 17–23.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In *Forty-first International Conference on Machine Learning*.

Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

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. In *Advances in Neural Information Processing Systems*, volume 36, pages 27469–27500. Curran Associates, Inc.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. BLOOM: A 176b-parameter open-access multilingual language model.

Bo Li, Jiaxin Ju, Can Wang, and Shirui Pan. 2023. How does ChatGPT affect fake news detection systems? In *Advanced Data Mining and Applications*, pages 565–580, Cham. Springer Nature Switzerland.

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

Yikang Liu, Ziyin Zhang, Wanyang Zhang, Shisen Yue, Xiaojing Zhao, Xinyuan Cheng, Yiwen Zhang, and Hai Hu. 2023. ArguGPT: evaluating, understanding and identifying argumentative essays generated by gpt models. *Preprint*, arXiv:2304.07666.

Ning Lu, Shengcai Liu, Rui He, Yew-Soon Ong, Qi Wang, and Ke Tang. 2024. Large language models can be guided to evade AI-generated text detection. *Transactions on Machine Learning Research*.

Durjoy Majumdar, Arjun S, Pranavi Boyina, Sri Sai Priya Rayidi, Yerra Rahul Sai, and Suryakanth V Gangashetty. 2024. Beyond text: Nefarious actors harnessing LLMs for strategic advantage. In *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, pages 1–7.

L. McInnes, J. Healy, and J. Melville. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Piotr Molenda, Adian Liusie, and Mark Gales. 2024. WaterJudge: Quality-detection trade-off when watermarking large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3515–3525, Mexico City, Mexico. Association for Computational Linguistics.

Thi Thuy An Ngo. 2023. The perception by university students of the use of ChatGPT in education. *International Journal of Emerging Technologies in Learning (Online)*, 18(17):4.

Charlotte Nicks, Eric Mitchell, Rafael Rafailov, Archit Sharma, Christopher D Manning, Chelsea Finn, and Stefano Ermon. 2023. Language model detectors are easily optimized against. In *The Twelfth International Conference on Learning Representations*.

OpenAI. 2024a. Deprecations. https://platform.openai.com/docs/deprecations. Accessed: 2024-07-14.

OpenAI. 2024b. OpenAI tokenizer. https://platform.openai.com/tokenizer. Accessed: 2024-04-08.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Xinlin Peng, Ying Zhou, Ben He, Le Sun, and Yingfei Sun. 2023. Hiding the ghostwriters: An adversarial evaluation of AI-generated student essay detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10406–10419, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *J. Mach. Learn. Res.*, 21(1).

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2024. Can AI-generated text be reliably detected? *Preprint*, arXiv:2303.11156.

Glorin Sebastian. 2023. Do ChatGPT and other AI chatbots pose a cybersecurity risk?: An exploratory study. *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)*, 15(1):1–11.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release Strategies and the Social Impacts of Language Models. *Preprint*, arXiv:1908.09203.

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

Miriam Sullivan, Andrew Kelly, and Paul McLaughlan. 2023. ChatGPT in higher education: Considerations for academic integrity and student learning. *Journal of Applied Learning & Teaching*, 6(1).

Burhan Ul Tayyab. 2023. Detect-GPT. https://github.com/BurhanUlTayyab/DetectGPT.

Undetectable. 2024. Undetectable. https://undetectable.ai. Accessed: 2024-10-08.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu,

Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant. 2018. Detecting homoglyph attacks with a siamese neural network. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 22–28.

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. 2023. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*.

Lixiang Yan, Lele Sha, Linxuan Zhao, Yuheng Li, Roberto Martinez-Maldonado, Guanliang Chen, Xinyu Li, Yueqiao Jin, and Dragan Gašević. 2024. Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology*, 55(1):90–112.

Peipeng Yu, Jiahan Chen, Xuan Feng, and Zhihua Xia. 2024. CHEAT: A large-scale dataset for detecting ChatGPT-writtEn AbsTracts. *Preprint*, arXiv:2304.12008.

Ruisi Zhang and Farinaz Koushanfar. 2024. EmMark: Robust watermarks for IP protection of embedded quantized large language models. *arXiv preprint arXiv:2402.17938*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Chaoyi Zhu, Jeroen Galjaard, Pin-Yu Chen, and Lydia Y Chen. 2024. Duwak: Dual watermarks in large language models. *arXiv preprint arXiv:2403.13000*.

Appendices

A Detection metrics

The detection metrics are reported in the following tables. The metrics include the MCC, accuracy, F1 score, precision and recall for each detector and dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.94	0.97	0.97	0.96	0.98
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.0	0.5	0.0	0.0	0.0

Table 3: Results for `ArguGPT` on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.93	0.96	0.96	0.94	0.99
5%	0.37	0.62	0.39	0.24	1.0
10%	0.11	0.51	0.05	0.03	1.0
15%	0.04	0.5	0.01	0.0	1.0
20%	0.02	0.5	0.0	0.0	1.0
Greedy	0.13	0.52	0.11	0.06	0.84

Table 4: Results for `Binoculars` on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.14	0.52	0.08	0.04	0.95
5%	-0.02	0.5	0.0	0.0	0.0
10%	0.03	0.51	0.13	0.08	0.55
15%	0.13	0.56	0.59	0.62	0.56
20%	0.06	0.52	0.64	0.86	0.51
Greedy	0.0	0.5	0.67	1.0	0.5

Table 5: Results for `DetectGPT` on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.9	0.95	0.95	0.94	0.95
5%	0.23	0.55	0.19	0.1	0.99
10%	0.04	0.5	0.01	0.0	1.0
15%	0.02	0.5	0.0	0.0	1.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	-0.01	0.5	0.13	0.07	0.48

Table 6: Results for `Fast-DetectGPT` on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.64	0.8	0.83	0.99	0.71
5%	0.41	0.69	0.63	0.53	0.79
10%	0.32	0.61	0.71	0.97	0.57
15%	0.12	0.52	0.67	1.0	0.51
20%	0.06	0.5	0.67	1.0	0.5
Greedy	0.02	0.5	0.67	1.0	0.5

Table 7: Results for Ghostbuster on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.47	0.7	0.61	0.46	0.9
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	-0.02	0.5	0.0	0.0	0.0
Greedy	0.0	0.5	0.0	0.0	0.0

Table 8: Results for OpenAI on the *CHEAT* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.92	0.96	0.96	0.95	0.96
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.0	0.5	0.0	0.0	0.0

Table 9: Results for ArguGPT on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.91	0.95	0.95	0.9	1.0
5%	0.22	0.55	0.17	0.1	1.0
10%	0.05	0.5	0.01	0.01	1.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.05	0.5	0.02	0.01	0.82

Table 10: Results for Binoculars on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.24	0.55	0.2	0.11	0.99
5%	-0.01	0.5	0.0	0.0	0.33
10%	0.11	0.53	0.16	0.09	0.72
15%	0.21	0.59	0.67	0.85	0.56
20%	0.08	0.51	0.67	0.99	0.51
Greedy	0.0	0.5	0.67	1.0	0.5

Table 11: Results for `DetectGPT` on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.88	0.94	0.94	0.89	0.99
5%	0.22	0.55	0.2	0.12	0.93
10%	0.04	0.5	0.01	0.0	1.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	-0.08	0.48	0.08	0.04	0.35

Table 12: Results for `Fast-DetectGPT` on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.92	0.96	0.96	0.95	0.96
5%	0.73	0.86	0.84	0.76	0.94
10%	0.51	0.71	0.77	0.99	0.64
15%	0.13	0.52	0.67	1.0	0.51
20%	0.0	0.5	0.67	1.0	0.5
Greedy	0.0	0.5	0.67	1.0	0.5

Table 13: Results for `Ghostbuster` on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	-0.21	0.43	0.09	0.06	0.22
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.5
Greedy	0.03	0.5	0.0	0.0	1.0

Table 14: Results for `OpenAI` on the *essay* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.92	0.96	0.96	0.95	0.97
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.0	0.5	0.0	0.0	0.0

Table 15: Results for ArguGPT on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.8	0.89	0.88	0.78	1.0
5%	0.22	0.55	0.17	0.09	1.0
10%	0.07	0.51	0.02	0.01	1.0
15%	0.03	0.5	0.0	0.0	1.0
20%	0.02	0.5	0.0	0.0	1.0
Greedy	0.08	0.51	0.03	0.02	0.94

Table 16: Results for Binoculars on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.23	0.56	0.22	0.12	0.93
5%	0.0	0.5	0.0	0.0	0.0
10%	0.03	0.5	0.01	0.0	0.8
15%	0.34	0.67	0.63	0.57	0.7
20%	0.14	0.54	0.67	0.94	0.52
Greedy	0.0	0.5	0.67	1.0	0.5

Table 17: Results for DetectGPT on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.92	0.96	0.96	0.92	1.0
5%	0.28	0.57	0.25	0.14	1.0
10%	0.1	0.51	0.04	0.02	1.0
15%	0.02	0.5	0.0	0.0	1.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.04	0.51	0.11	0.06	0.58

Table 18: Results for Fast-DetectGPT on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.93	0.96	0.96	0.94	0.99
5%	0.61	0.79	0.75	0.63	0.93
10%	0.51	0.72	0.78	0.98	0.64
15%	0.16	0.53	0.68	1.0	0.51
20%	0.04	0.5	0.67	1.0	0.5
Greedy	0.0	0.5	0.67	1.0	0.5

Table 19: Results for Ghostbuster on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.27	0.57	0.25	0.14	0.98
5%	0.0	0.5	0.0	0.0	0.0
10%	-0.04	0.5	0.0	0.0	0.0
15%	-0.09	0.49	0.0	0.0	0.0
20%	-0.11	0.49	0.0	0.0	0.06
Greedy	-0.06	0.5	0.0	0.0	0.0

Table 20: Results for OpenAI on the *reuter* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.39	0.63	0.42	0.26	1.0
5%	0.0	0.5	0.0	0.0	0.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	0.0	0.5	0.0	0.0	0.0

Table 21: Results for ArguGPT on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.85	0.92	0.91	0.84	1.0
5%	0.2	0.54	0.14	0.08	1.0
10%	0.0	0.5	0.0	0.0	0.0
15%	0.0	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.0
Greedy	-0.04	0.5	0.01	0.0	0.23

Table 22: Results for Binoculars on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.44	0.68	0.55	0.39	0.93
5%	0.04	0.5	0.01	0.0	1.0
10%	0.01	0.5	0.09	0.05	0.52
15%	0.02	0.51	0.57	0.65	0.51
20%	0.02	0.51	0.66	0.96	0.5
Greedy	0.0	0.5	0.67	1.0	0.5

Table 23: Results for `DetectGPT` on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.79	0.89	0.88	0.81	0.97
5%	0.3	0.59	0.31	0.19	0.96
10%	0.05	0.5	0.02	0.01	0.83
15%	-0.03	0.5	0.0	0.0	0.0
20%	0.0	0.5	0.0	0.0	0.5
Greedy	-0.33	0.37	0.09	0.07	0.17

Table 24: Results for `Fast-DetectGPT` on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.88	0.94	0.94	0.9	0.98
5%	0.42	0.66	0.49	0.33	0.96
10%	0.64	0.82	0.83	0.86	0.79
15%	0.33	0.6	0.71	0.99	0.56
20%	0.09	0.51	0.67	1.0	0.5
Greedy	0.0	0.5	0.67	1.0	0.5

Table 25: Results for `Ghostbuster` on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	-0.05	0.5	0.0	0.0	0.11
5%	-0.04	0.5	0.0	0.0	0.0
10%	-0.05	0.5	0.0	0.0	0.0
15%	-0.13	0.48	0.0	0.0	0.0
20%	-0.11	0.49	0.01	0.0	0.11
Greedy	0.01	0.5	0.01	0.0	0.57

Table 26: Results for `OpenAI` on the *writing prompts* dataset.

Type	MCC	Accuracy	F1	Precision	Recall
Original	0.92	0.96	0.96	0.95	0.96
5%	0.18	0.54	0.14	0.08	0.94
10%	-0.01	0.5	0.01	0.0	0.43
15%	0.0	0.5	0.01	0.0	0.5
20%	-0.03	0.5	0.0	0.0	0.29
Greedy	0.0	0.5	0.01	0.01	0.5
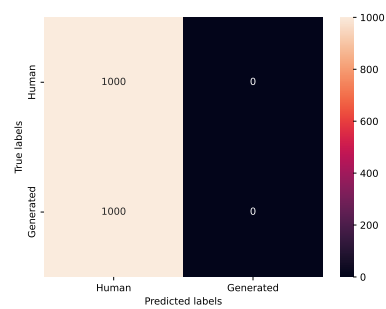
Table 27: Results for `Watermark` on the *realnewslike* dataset.

B Confusion matrices
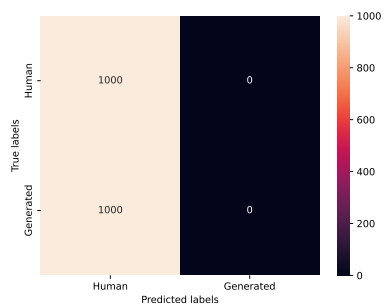
The following figures show the confusion matrices for each detector and dataset. As stated in Section 2.3, the datasets used in the experiments are *CHEAT*, *essay*, *reuter*, *writing prompts*, and *realnewslike* (only used for the watermarking detector). Each dataset contains 1,000 human-written examples and 1,000 AI-written examples.
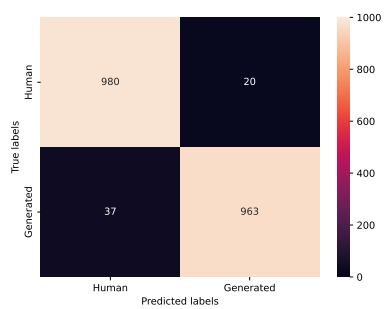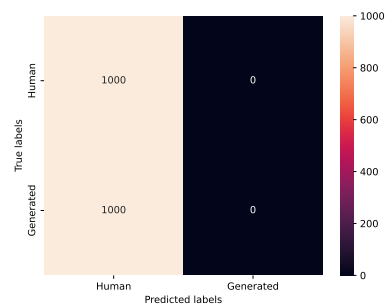
(a) No attack
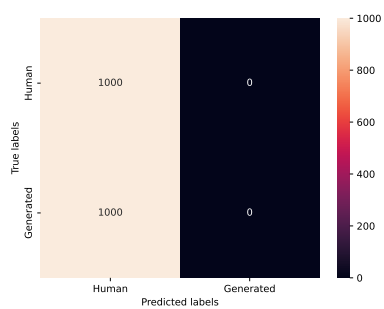
(b) Random attack (5%)
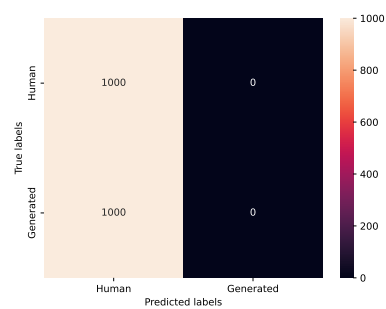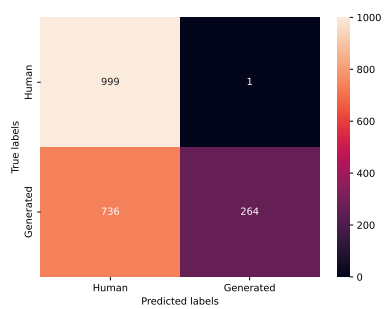
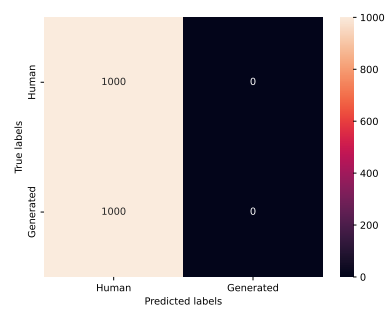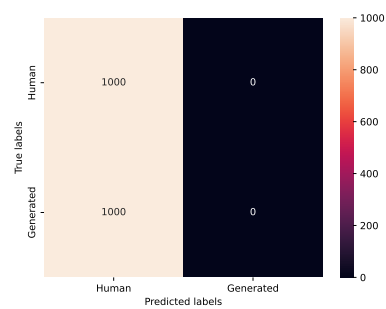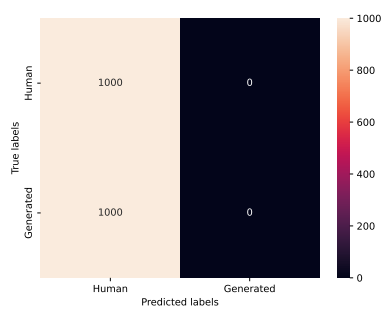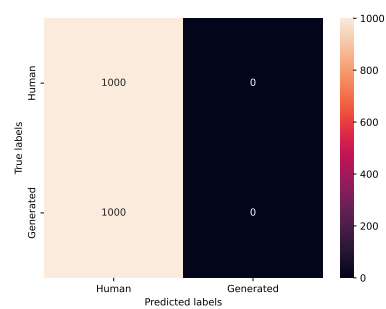(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 5: Confusion matrices for the ArguGPT detector on the *CHEAT* dataset.
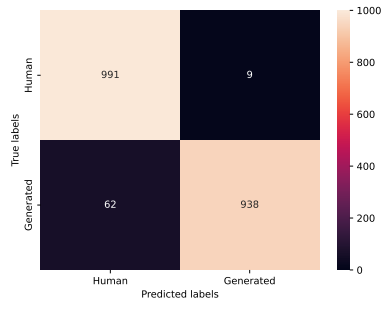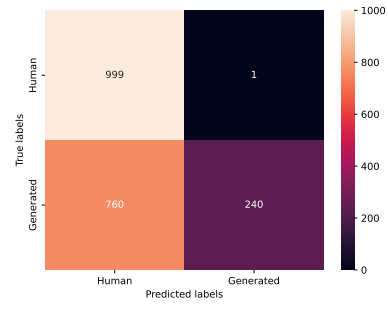
(a) No attack
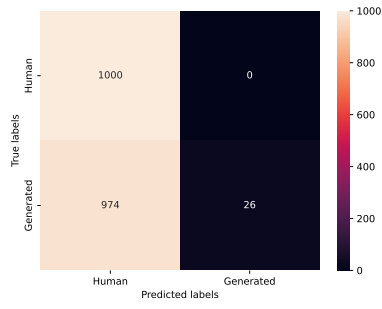
(b) Random attack (5%)
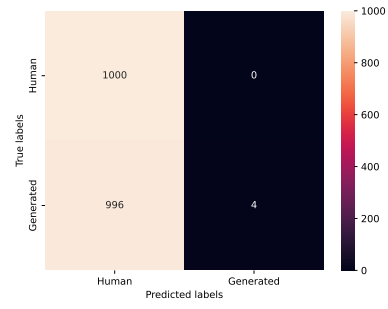
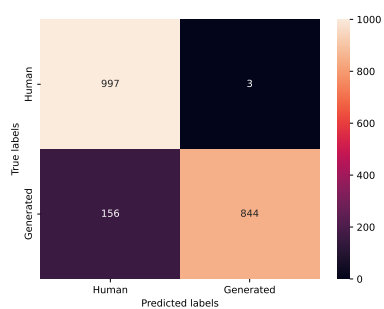(c) Random attack (10%)
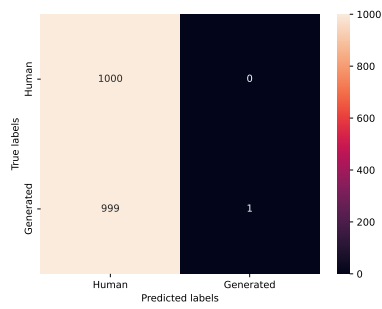
(d) Random attack (15%)
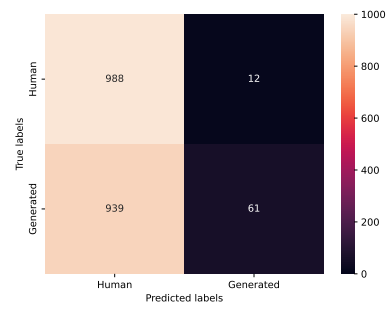
(e) Random attack (20%)

(f) Greedy attack

Figure 6: Confusion matrices for the `Binoculars` detector on the *CHEAT* dataset.
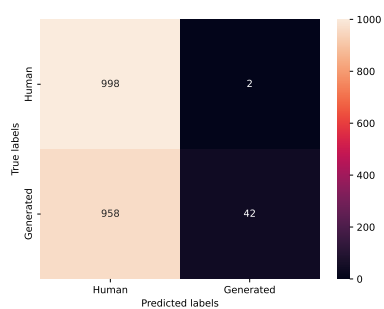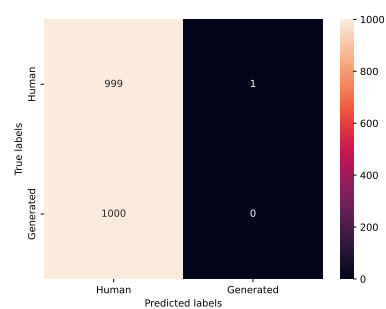
(a) No attack
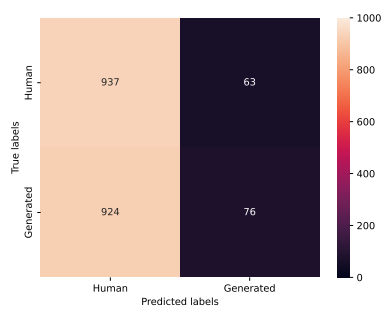
(b) Random attack (5%)

(c) Random attack (10%)
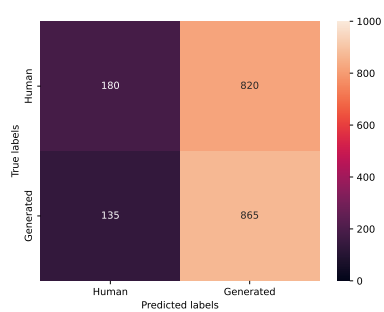
(d) Random attack (15%)
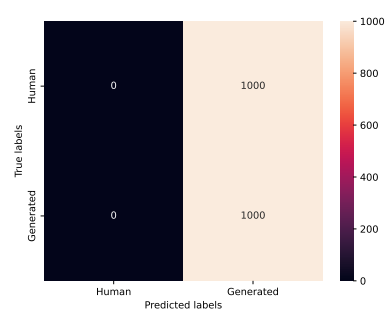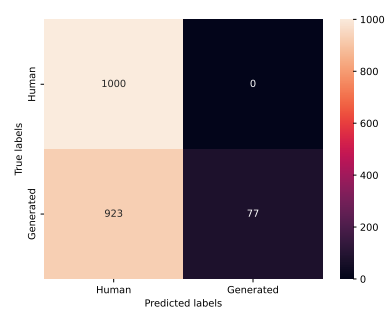
(e) Random attack (20%)
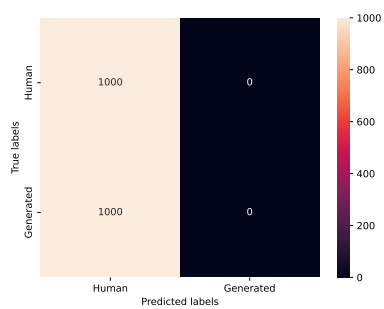
(f) Greedy attack

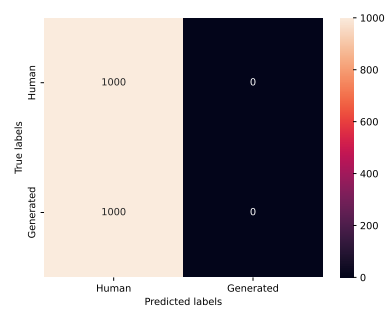Figure 7: Confusion matrices for DetectGPT on the *CHEAT* dataset.
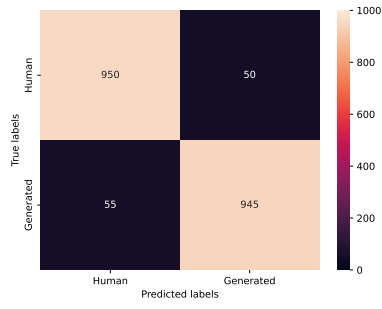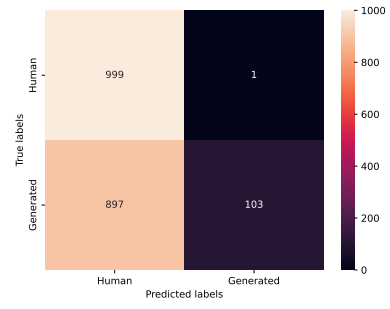
(a) No attack
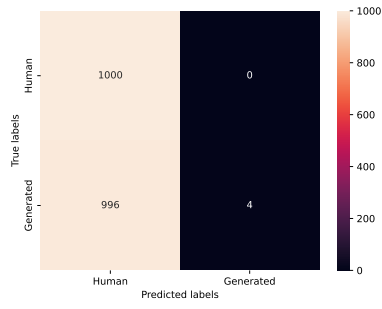
(b) Random attack (5%)
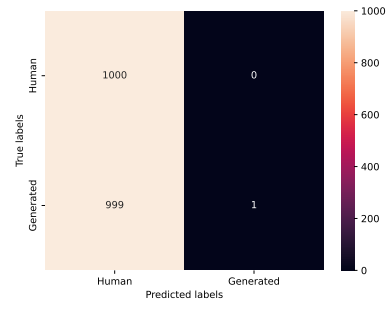
(c) Random attack (10%)
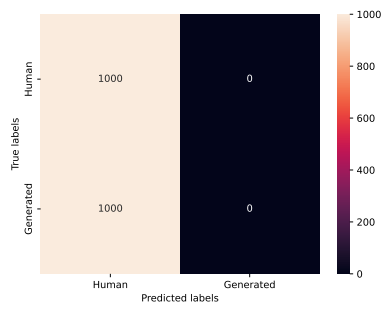
(d) Random attack (15%)
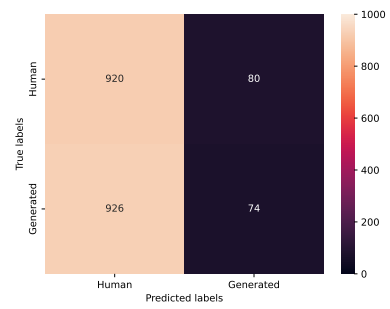
(e) Random attack (20%)

(f) Greedy attack

Figure 8: Confusion matrices for the Fast-DetectGPT detector on the *CHEAT* dataset.
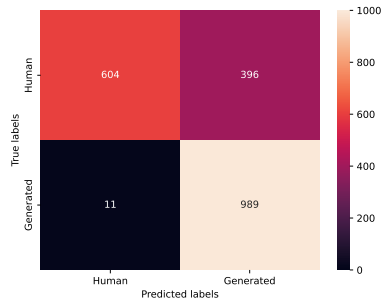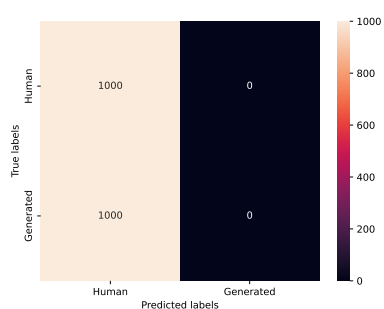
(a) No attack
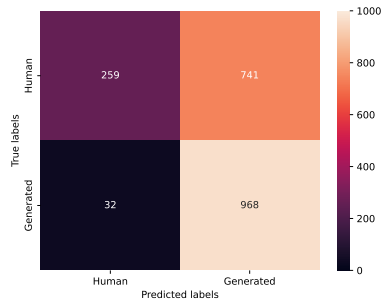
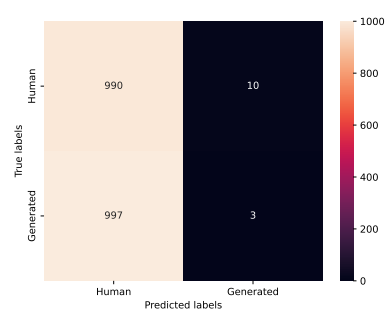(b) Random attack (5%)
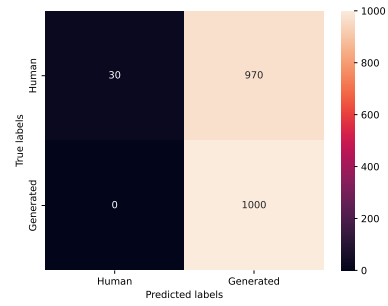
(c) Random attack (10%)
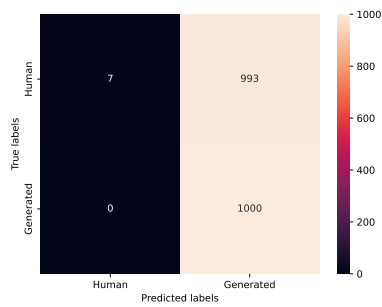
(d) Random attack (15%)
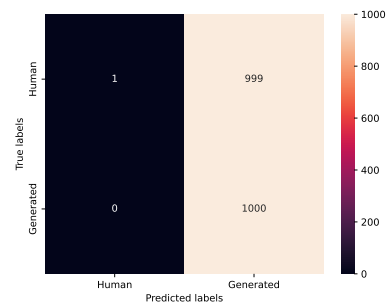
(e) Random attack (20%)

(f) Greedy attack

Figure 9: Confusion matrices for the Ghostbuster detector on the *CHEAT* dataset.
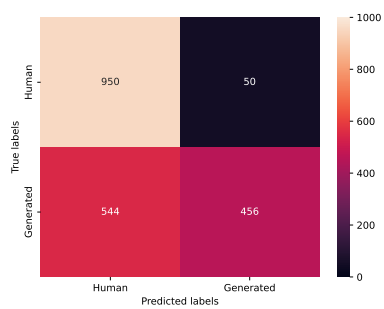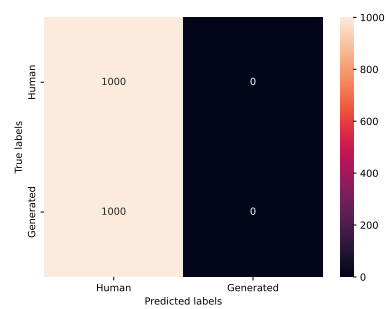
(a) No attack
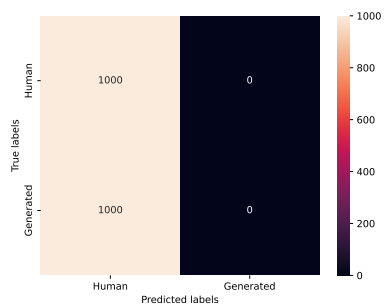
(b) Random attack (5%)
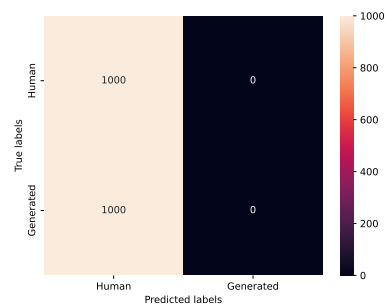
(c) Random attack (10%)
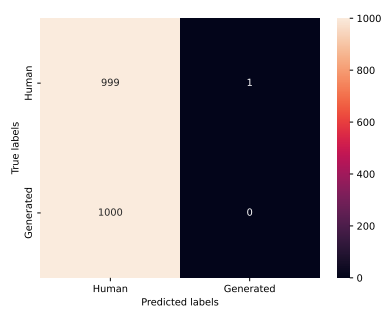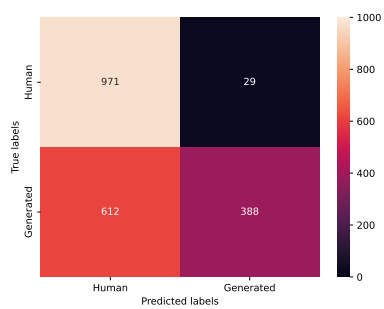
(d) Random attack (15%)
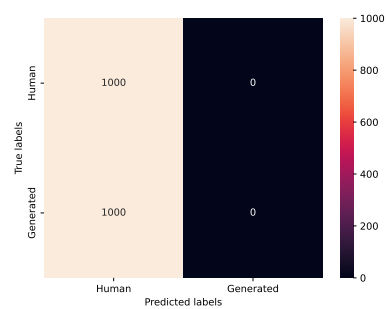
(e) Random attack (20%)

(f) Greedy attack

Figure 10: Confusion matrices for the `OpenAI` detector on the *CHEAT* dataset.
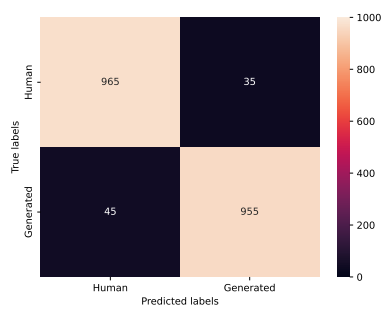
(a) No attack

(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)
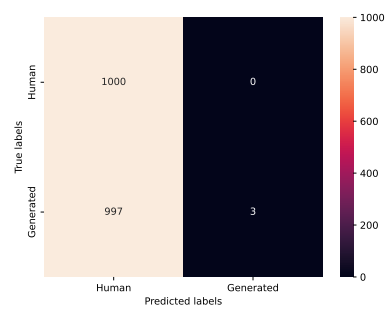
(e) Random attack (20%)
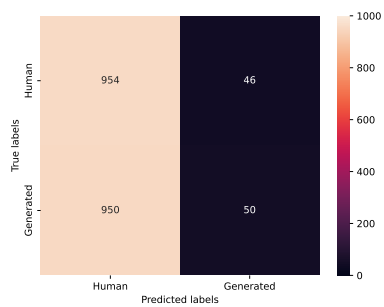
(f) Greedy attack

Figure 11: Confusion matrices for the ArguGPT detector on the *essay* dataset.
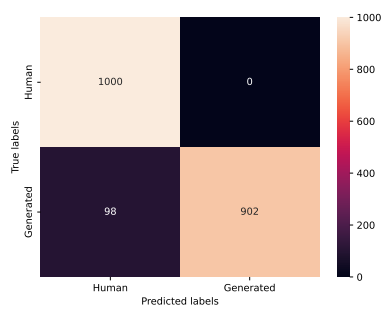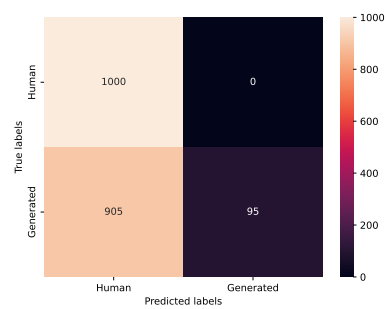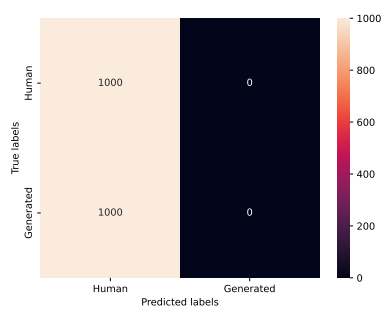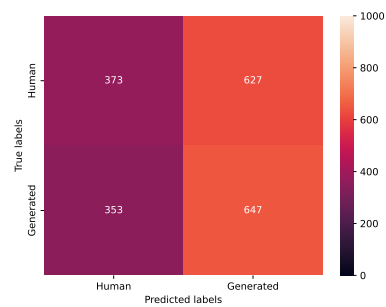
(a) No attack
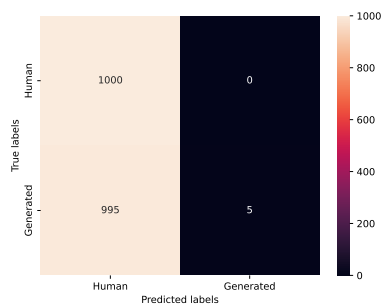
(b) Random attack (5%)
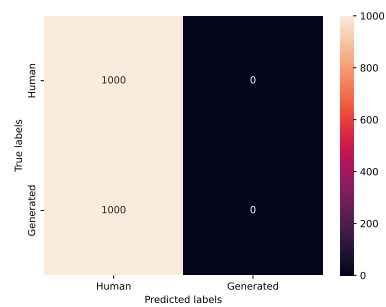
(c) Random attack (10%)

(d) Random attack (15%)
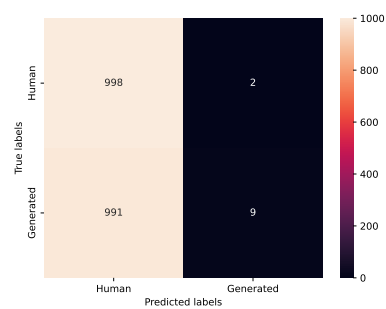
(e) Random attack (20%)

(f) Greedy attack

Figure 12: Confusion matrices for the `Binoculars` detector on the *essay* dataset.
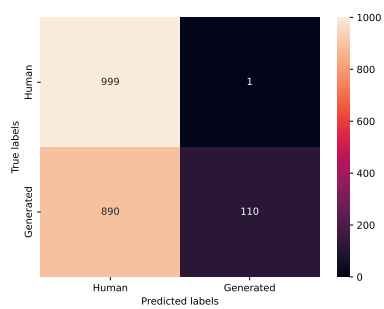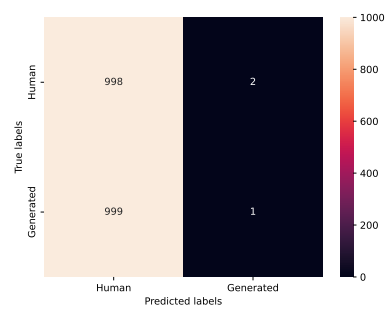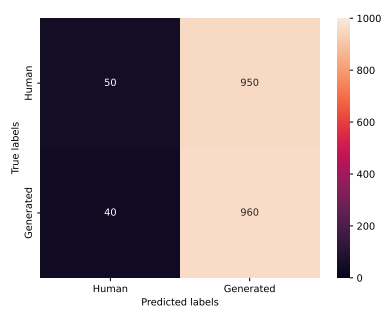
(a) No attack
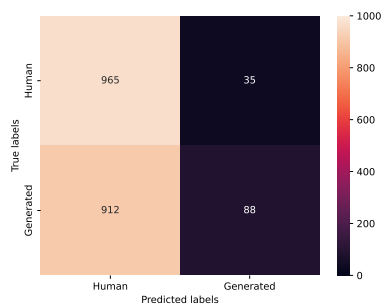
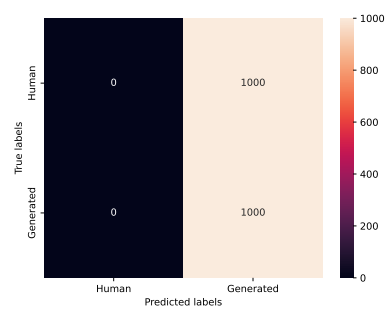(b) Random attack (5%)

(c) Random attack (10%)
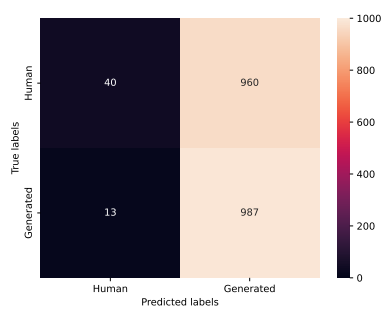
(d) Random attack (15%)
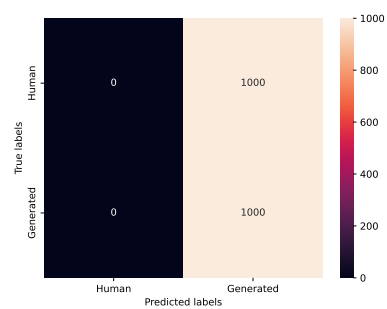
(e) Random attack (20%)

(f) Greedy attack

Figure 13: Confusion matrices for DetectGPT on the *essay* dataset.
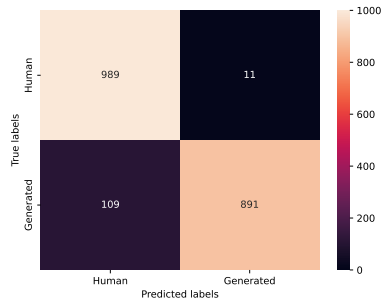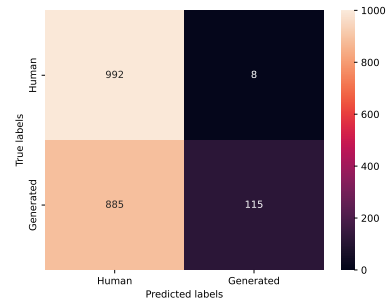
(a) No attack
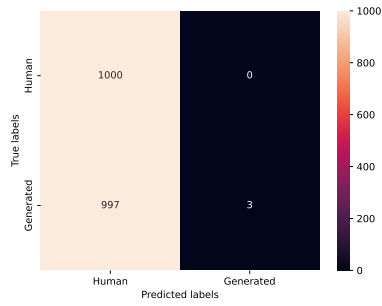
(b) Random attack (5%)
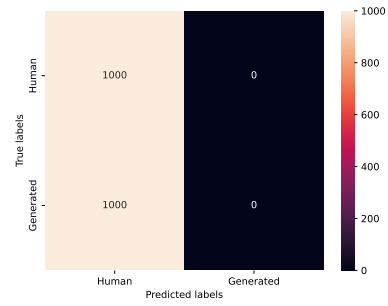
(c) Random attack (10%)
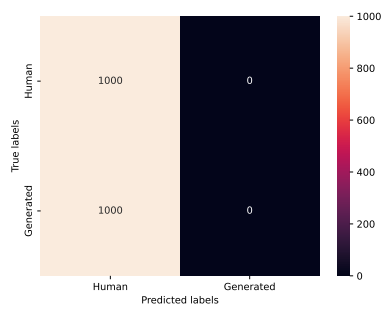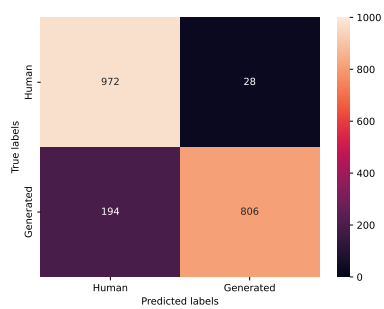
(d) Random attack (15%)
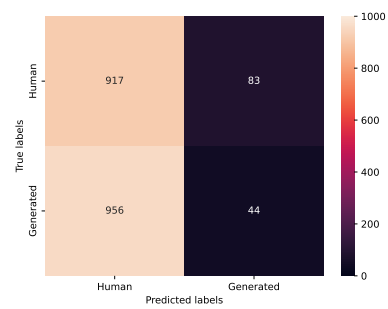
(e) Random attack (20%)

(f) Greedy attack

Figure 14: Confusion matrices for the `Fast-DetectGPT` detector on the *essay* dataset.
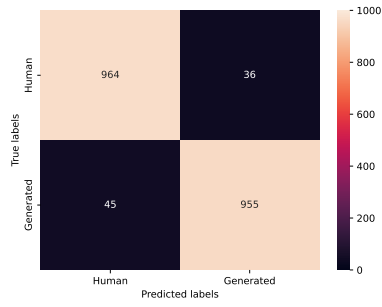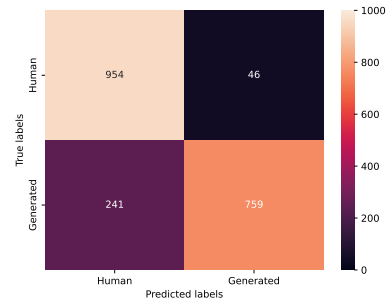
(a) No attack
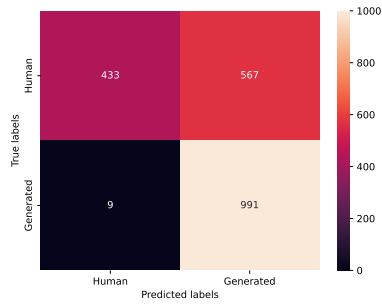
(b) Random attack (5%)
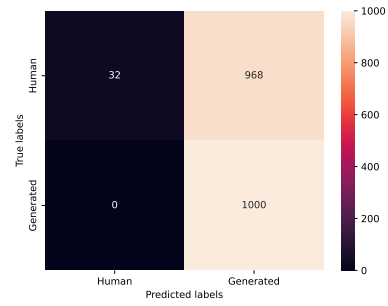
(c) Random attack (10%)
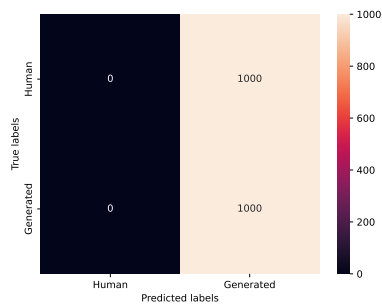
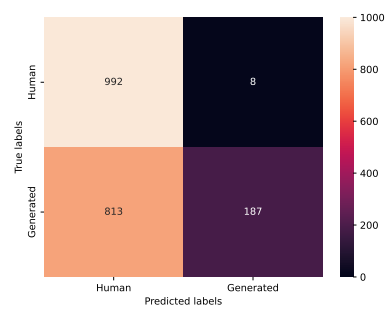(d) Random attack (15%)
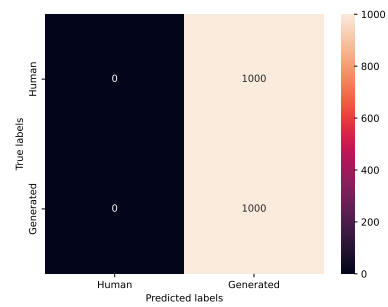
(e) Random attack (20%)
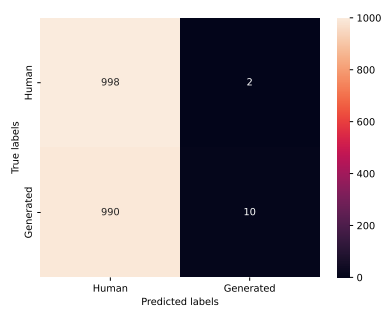
(f) Greedy attack

Figure 15: Confusion matrices for the `Ghostbuster` detector on the *essay* dataset.
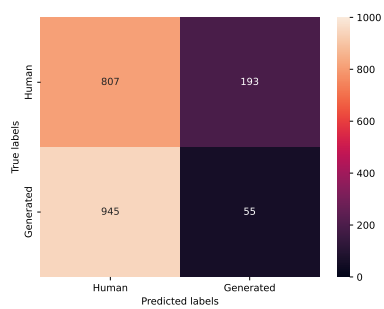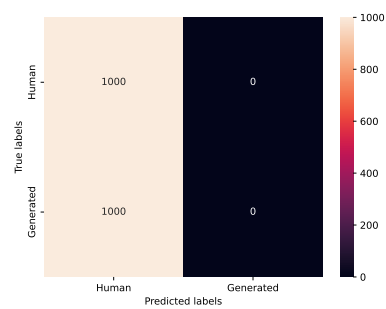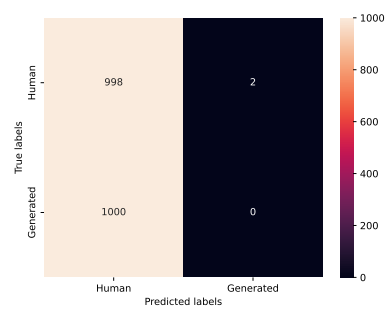
(a) No attack
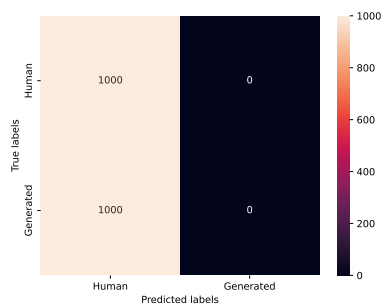
(b) Random attack (5%)
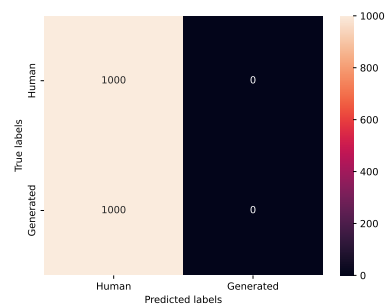
(c) Random attack (10%)
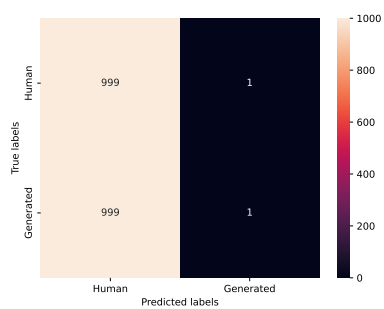
(d) Random attack (15%)
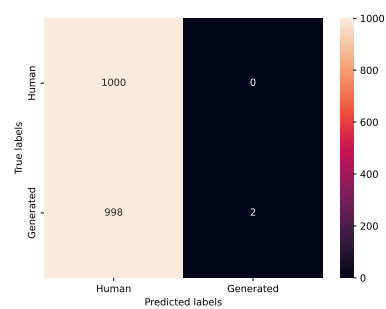
(e) Random attack (20%)

(f) Greedy attack

Figure 16: Confusion matrices for the `OpenAI` detector on the *essay* dataset.
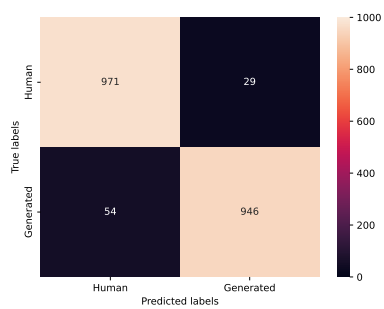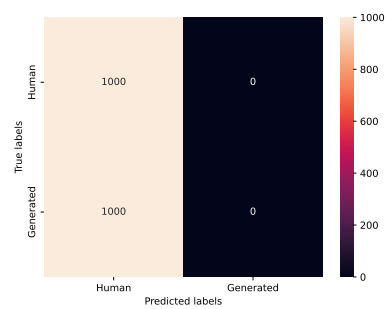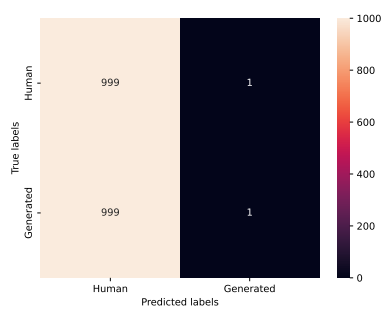
(a) No attack
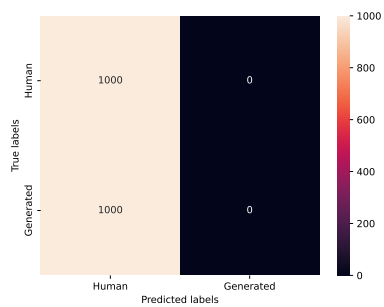
(b) Random attack (5%)
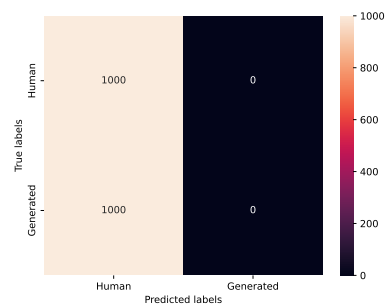
(c) Random attack (10%)
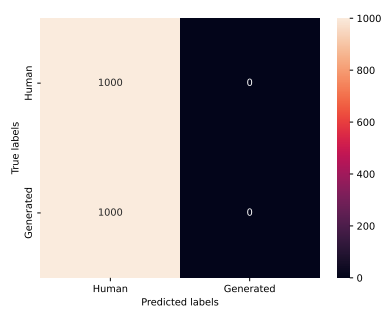
(d) Random attack (15%)
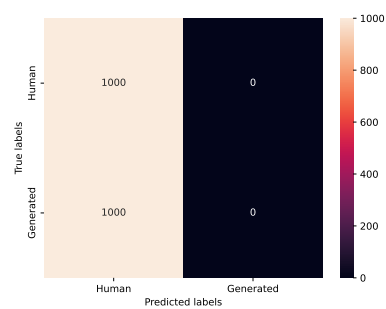
(e) Random attack (20%)

(f) Greedy attack

Figure 17: Confusion matrices for the `ArguGPT` detector on the *reuter* dataset.
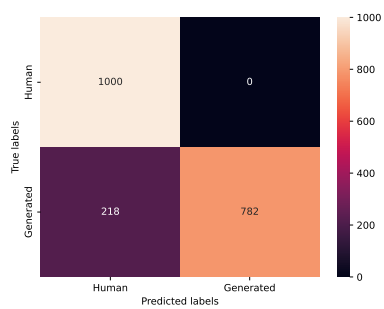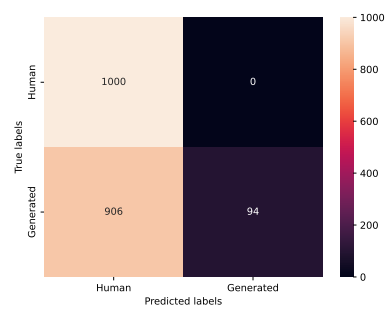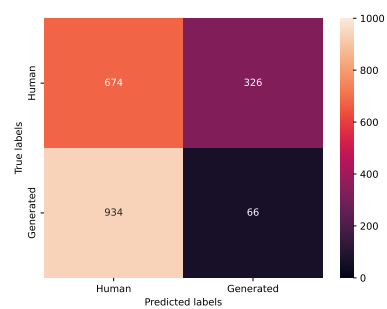
(a) No attack
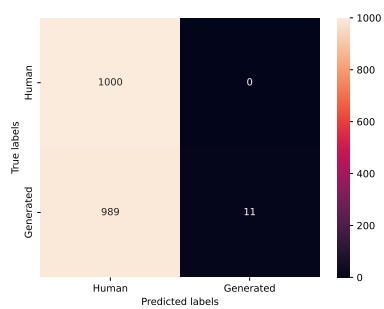
(b) Random attack (5%)
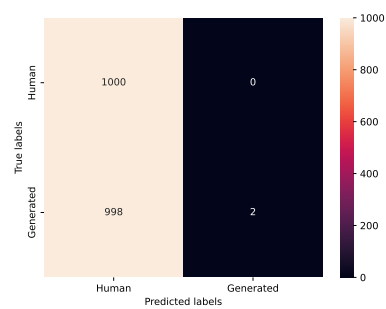
(c) Random attack (10%)
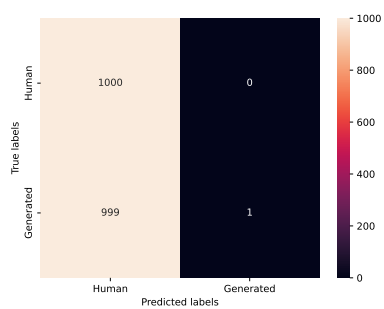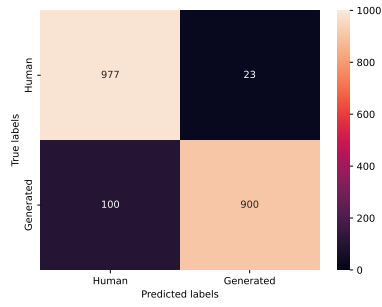
(d) Random attack (15%)
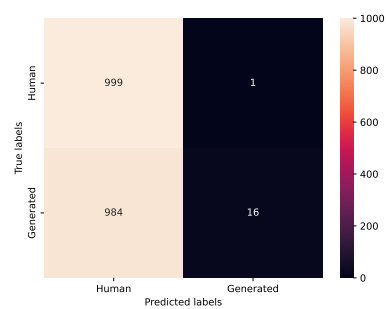
(e) Random attack (20%)

(f) Greedy attack

Figure 18: Confusion matrices for the `Binoculars` detector on the *reuter* dataset.
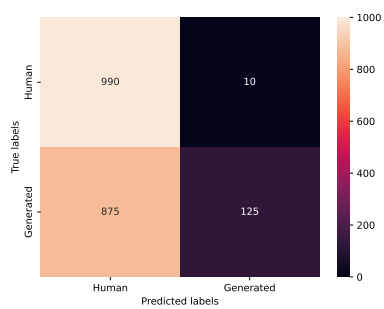
(a) No attack
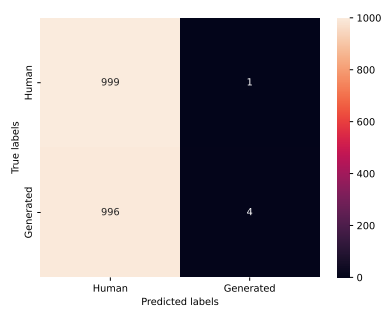
(b) Random attack (5%)
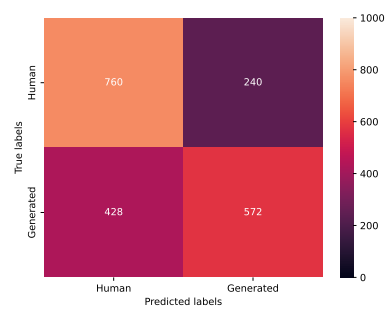
(c) Random attack (10%)
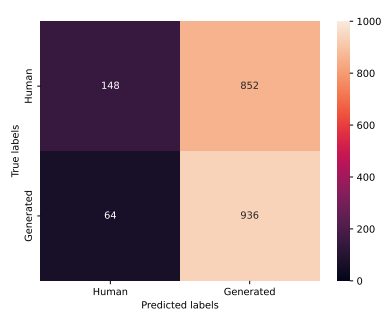
(d) Random attack (15%)
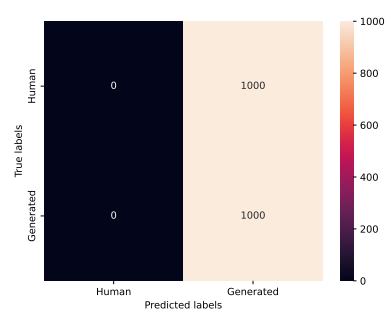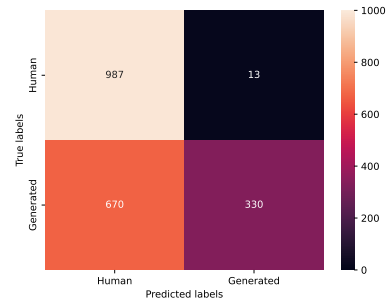
(e) Random attack (20%)

(f) Greedy attack

Figure 19: Confusion matrices for DetectGPT on the *reuter* dataset.
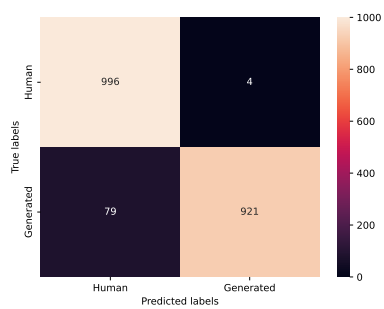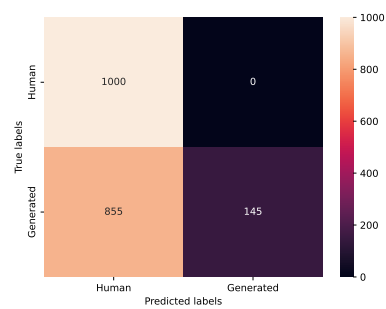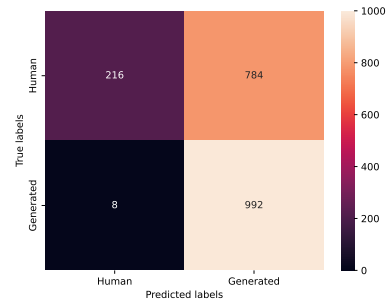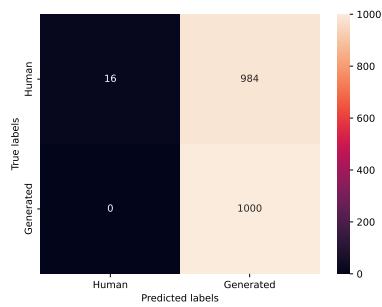
(a) No attack
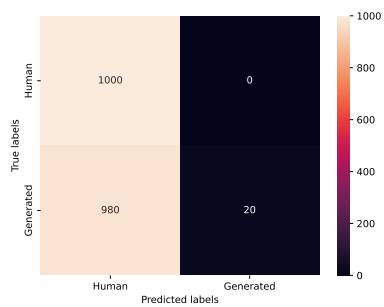
(b) Random attack (5%)
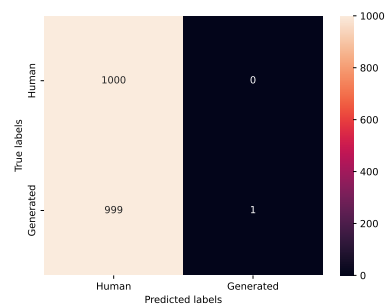
(c) Random attack (10%)

(d) Random attack (15%)
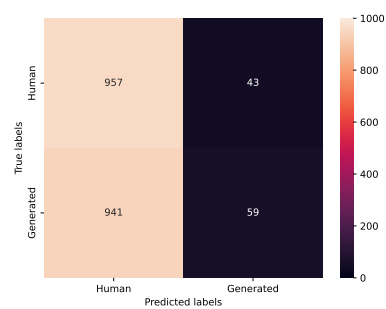
(e) Random attack (20%)

(f) Greedy attack

Figure 20: Confusion matrices for the Fast-DetectGPT detector on the *reuter* dataset.
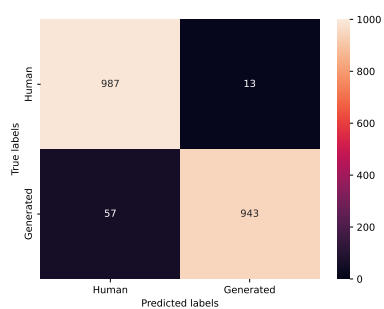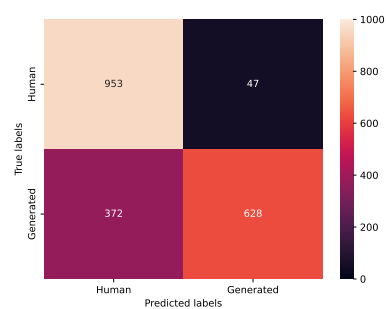
(a) No attack
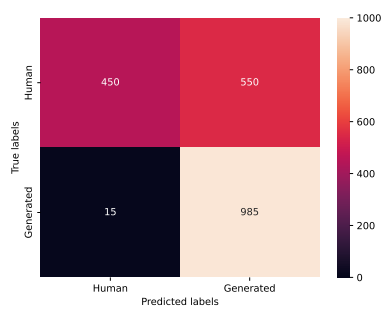
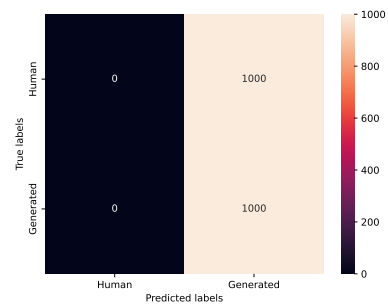(b) Random attack (5%)
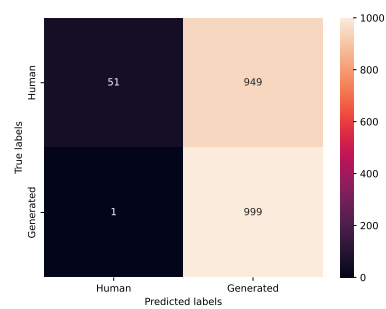
(c) Random attack (10%)
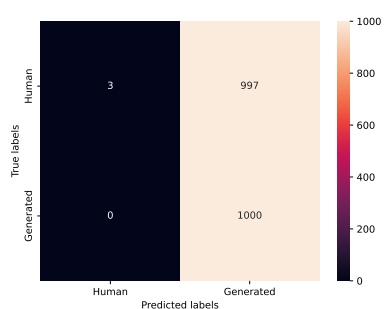
(d) Random attack (15%)
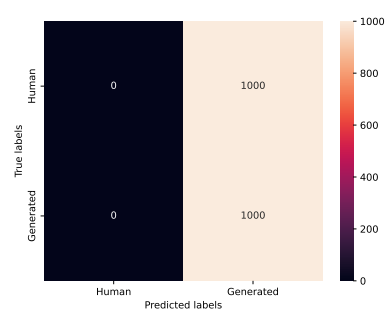
(e) Random attack (20%)

(f) Greedy attack

Figure 21: Confusion matrices for the Ghostbuster detector on the *reuter* dataset.
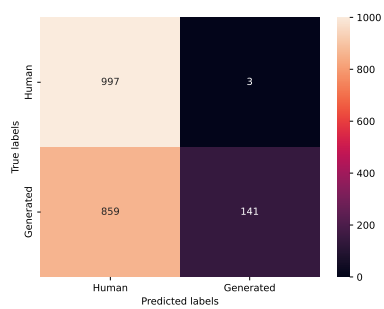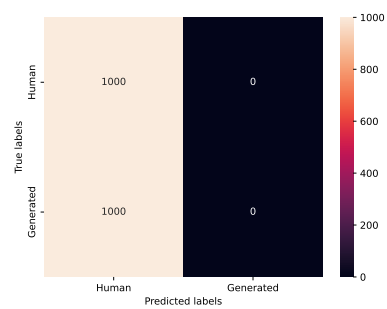
(a) No attack
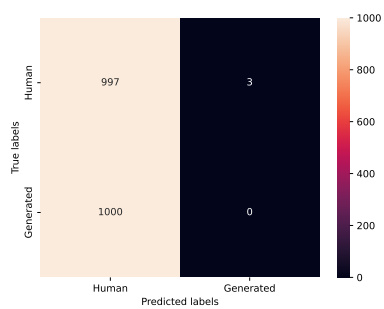
(b) Random attack (5%)
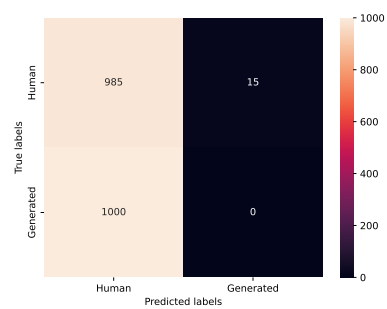
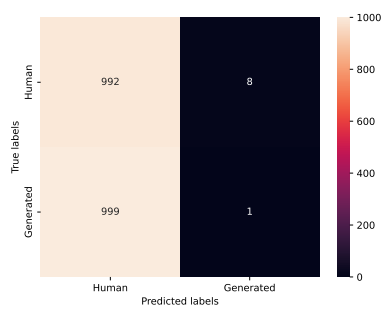(c) Random attack (10%)
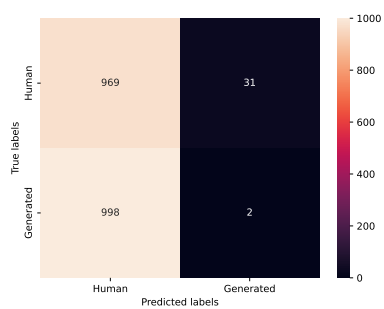
(d) Random attack (15%)
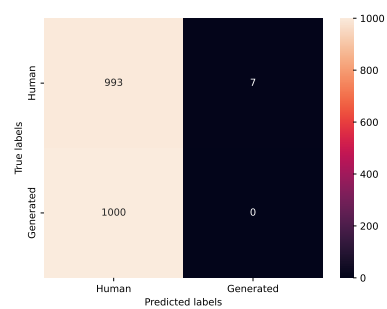
(e) Random attack (20%)

(f) Greedy attack

Figure 22: Confusion matrices for the `OpenAI` detector on the *reuter* dataset.

(a) No attack
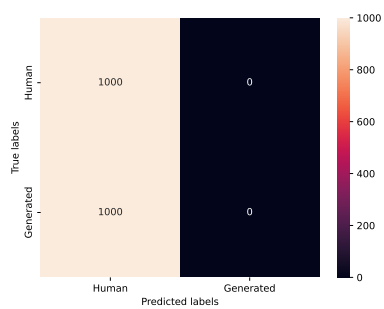
(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)
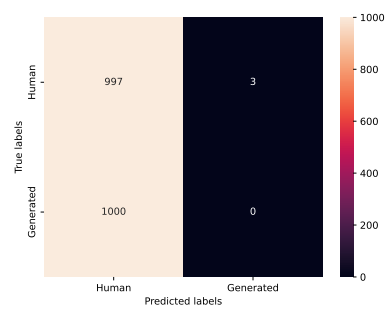
(e) Random attack (20%)
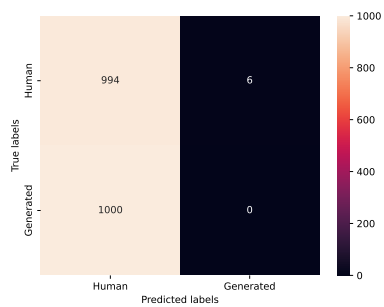
(f) Greedy attack

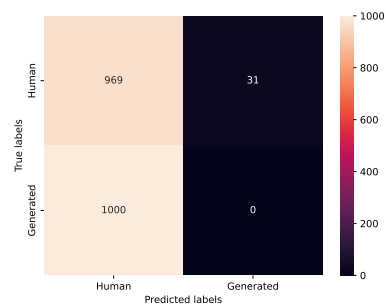Figure 23: Confusion matrices for the ArguGPT detector on the *writing prompts* dataset.

(a) No attack

(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 24: Confusion matrices for the `Binoculars` detector on the *writing prompts* dataset.
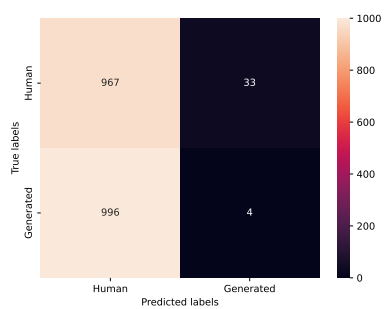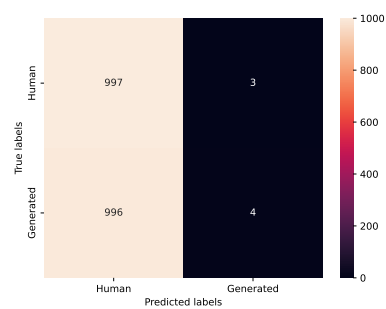
(a) No attack

(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 25: Confusion matrices for `DetectGPT` on the *writing prompts* dataset.

(a) No attack

(b) Random attack (5%)

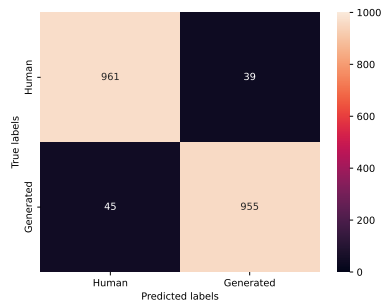(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 26: Confusion matrices for the `Fast-DetectGPT` detector on the *writing prompts* dataset.

(a) No attack

(b) Random attack (5%)

(c) Random attack (10%)
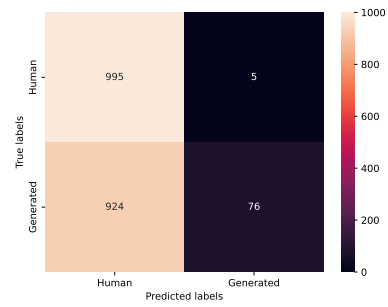
(d) Random attack (15%)

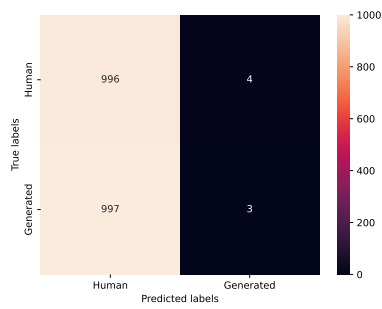(e) Random attack (20%)

(f) Greedy attack

Figure 27: Confusion matrices for the Ghostbuster detector on the *writing prompts* dataset.
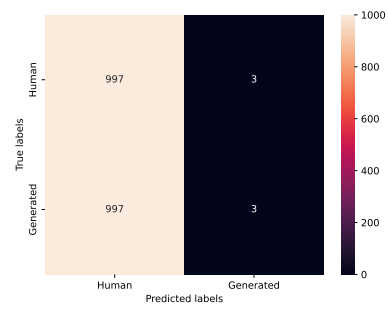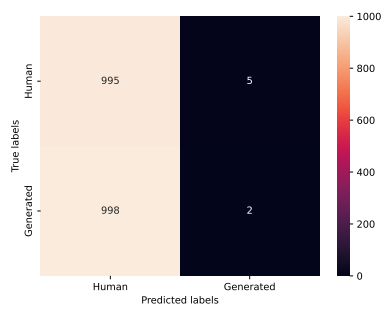
(a) No attack

(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 28: Confusion matrices for the `OpenAI` detector on the *writing prompts* dataset.

(a) No attack
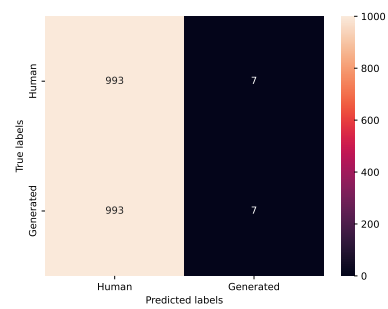
(b) Random attack (5%)

(c) Random attack (10%)

(d) Random attack (15%)

(e) Random attack (20%)

(f) Greedy attack

Figure 29: Confusion matrices for the watermarking-based detector on the *realnewslike* dataset. Here, "generated" refers to the watermarked versions of the texts.