OUTFOX: LLM-Generated Essay Detection Through In-Context Learning with Adversarially Generated Examples

Ryuto Koike[1], **Masahiro Kaneko**[2,1], **Naoaki Okazaki**[1]
[1]Tokyo Institute of Technology
[2]MBZUAI
ryuto.koike@nlp.c.titech.ac.jp, masahiro.kaneko@mbzuai.ac.ae, okazaki@c.titech.ac.jp

Abstract

Large Language Models (LLMs) have achieved human-level fluency in text generation, making it difficult to distinguish between human-written and LLM-generated texts. This poses a growing risk of misuse of LLMs and demands the development of detectors to identify LLM-generated texts. However, existing detectors lack robustness against attacks: they degrade detection accuracy by simply paraphrasing LLM-generated texts. Furthermore, a malicious user might attempt to deliberately evade the detectors based on detection results, but this has not been assumed in previous studies. In this paper, we propose **OUTFOX**, a framework that improves the robustness of LLM-generated-text detectors by allowing both the detector and the attacker to consider each other's output. In this framework, the attacker uses the detector's prediction labels as examples for in-context learning and adversarially generates essays that are harder to detect, while the detector uses the adversarially generated essays as examples for in-context learning to learn to detect essays from a strong attacker. Experiments in the domain of student essays show that the proposed detector improves the detection performance on the attacker-generated texts by up to +41.3 points F1-score. Furthermore, the proposed detector shows a state-of-the-art detection performance: up to 96.9 points F1-score, beating existing detectors on non-attacked texts. Finally, the proposed attacker drastically degrades the performance of detectors by up to -57.0 points F1-score, massively outperforming the baseline paraphrasing method for evading detection.

Introduction

LLMs, characterized by their enormous model size and vast training data, have demonstrated emergent abilities with impressive performance across various tasks (Wei et al. 2022). These abilities include a high degree of language comprehension, fluent generation, and the capacity to handle tasks unseen during training through in-context learning (Brown et al. 2020; Ouyang et al. 2022; Sanh et al. 2022).

Despite these successes, there are growing concerns about the potential misuse of LLMs. A notable example is in education, where students might copy and paste text generated by LLMs, such as ChatGPT (OpenAI 2023c), for their assignments. This concern has led to the development of detectors designed to identify LLM-generated text with promising detection performance (Kirchenbauer et al. 2023; Mitchell et al. 2023; OpenAI 2023a; Tang, Chuang, and Hu 2023).

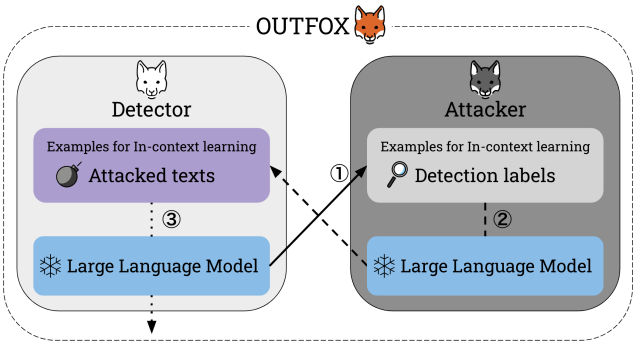

Figure 1: In our OUTFOX framework, there are three steps. Step ①: The detector outputs prediction labels to texts in a training set. Step ②: The attacker uses the detector's prediction labels as examples for in-context learning to generate more sophisticated attacks against a training set. Step ③: The detector uses these adversarially generated texts by a strong attacker to detect texts in a test set.

Unfortunately, existing detectors often perform poorly against simple attacks (e.g., paraphrasing), as highlighted by recent studies (Sadasivan et al. 2023; Krishna et al. 2023). A recent survey called for developing robust detection methods against other potential attacks designed to deceive the detectors (Tang, Chuang, and Hu 2023). Given the human-like generative abilities of LLMs, there's the unexplored risk that malicious users might exploit LLMs to create texts specifically designed to evade detection.

Motivated by this need, we propose **OUTFOX**, a novel framework designed to enhance the robustness and applicability of LLM-generated text detectors. As illustrated in Figure 1, OUTFOX introduces an approach where both the detector and the attacker learn from each other's output. The attacker uses the detector's predictions as examples for in-context learning to generate more sophisticated attacks. In contrast, the detector uses these adversarially generated texts as examples for in-context learning to improve its detection abilities against a strong attacker. To validate our approach in a realistic domain, we create a dataset of native-speaker student essay writing to detect LLM-generated essays. Our dataset contains 15,400 triplets of essay problem statements,

student-written essays, and LLM-generated essays.

Experiments show that our OUTFOX detector substantially improves the detection performance on the attacker-generated texts by up to +41.3 points F1-score compared to without considering attacks. This result empirically suggests that LLMs, especially ChatGPT, might learn implicit differences between non-attacked and attacked texts via in-context examples. Interestingly, our OUTFOX detector performs consistently as well or even better on non-attacked texts, resulting in the state-of-the-art detection performance of up to 96.9 points F1-score on non-attacked texts. This result demonstrates that considering attacks in our detector has little negative effect on detection performance on non-attacked texts. Furthermore, our OUTFOX attacker can drastically degrade the performance of detectors, with a decrease of up to -57.0 points F1-score, which massively outperforms the baseline paraphrasing method for evading detection. In our analysis, we explore the semantic differences between non-attacked and attacked essays. The analysis reveals that our attacker-generated essays are semantically closer to human-written essays than non-attacked essays, leading to success in such effective attacking. For reproducibility, we release our code and dataset.[1]

Related Work

LLM-Generated Text Detection Tackling the malicious uses of LLMs, recent studies have proposed detectors to identify LLM-generated texts. These detectors can be mainly categorized into watermarking algorithms, statistical outlier detection methods, and supervised classifiers. Watermarking algorithms use token-level secret markers in texts that humans cannot recognize for detection (Kirchenbauer et al. 2023). In order to embed the markers into texts, the probabilities of selected tokens by a hash function are modified to be higher in text generation at each time step. Our work focuses not on the watermark-enhanced LMs, but on LLMs that are openly used in our daily life—hence our chosen domain of student essays. Statistical outlier detection methods exploit statistical differences in linguistic features between human-written and LLM-generated texts. These includes n-gram frequencies (Badaskar, Agarwal, and Arora 2008), entropy (Lavergne, Urvoy, and Yvon 2008), perplexity (Beresneva 2016), token log probabilities (Solaiman et al. 2019), token ranks (Ippolito et al. 2020), and negative curvature regions of the model's log probability (Mitchell et al. 2023). Supervised classifiers are models specifically trained to discriminate human-written and LLM-generated texts with labels. The classifiers range from classical methods (Ippolito et al. 2020; Crothers, Japkowicz, and Viktor 2023) to neural-based methods (Solaiman et al. 2019; Bakhtin et al. 2019; Uchendu et al. 2020; Rodriguez et al. 2022; Guo et al. 2023).

Detection for Assessing Academic Dishonesty In the educational context, LLM-based services (e.g., ChatGPT and Bard (Google 2023)) have the potential to help students with their writing. However, many schools consider these services as academic dishonesty because students can misuse these tools to cheat on assignments (OpenAI 2023b). Based on this, recent studies have investigated LLM-generated text detection for student assignments, such as argumentative essays (Liu et al. 2023) and university-level course problems (Ibrahim et al. 2023; Vasilatos et al. 2023). Specifically, Liu et al. (2023) targeted the essay domain written by non-native English learners. However, considering the human-level generative capabilities of LLMs, it can be more difficult to classify LLM-generated and native-written essays than non-native-written ones. Thus, as a more challenging setting, we create a dataset to distinguish native-student-written from LLM-generated essays.

Attacking LLM-Generated Text Detection Most recent studies have reported the effectiveness of paraphrasing attacks, where existing detectors experience a large loss in accuracy when given a paraphrased version of an LLM-generated text (Sadasivan et al. 2023; Krishna et al. 2023). For instance, Krishna et al. (2023) proposed DIPPER, the 11B document-level paraphraser, controlling output diversity on vocabulary and content re-ordering. Considering the human-level generative abilities of LLMs, malicious users might instruct LLMs to adversarially generate texts based on detection results, but this has not been explored in prior work.

Defense against Attacking LLM-Generated Text Detection The existence of these attacking methods, in turn, poses a need for a defense against such attacks. There are few studies specifically about defending against the attacks. Krishna et al. (2023) proposed a retrieval-based defense that detects a text that is semantically similar to one of the LLM-generated responses in an API database. However, it needs active actions by API providers, and based on the nature of the method, a false positive rate can be higher with a larger database. In addition, Sadasivan et al. (2023) recently showed that the retrieval-based defense method is vulnerable against recursive paraphrasing. The detection accuracy significantly drops to 25% after 5 rounds of paraphrasing by Krishna et al. (2023)'s paraphraser.

In recent concurrent work by Hu, Chen, and Ho (2023), they proposed RADAR, a framework that jointly trains a robust LLM-generated-text classifier against the paraphrasing attack via adversarial learning. In our OUTFOX framework, the detector and the attacker can learn from each other's adversarially generated output via in-context learning, without any parameter updates. Thus, the detector in our framework can handle new attacks by simply adding these to in-context examples without additional fine-tuning. Additionally, the attacker in our framework is not limited to the paraphrasing attack but is directly designed to deceive the detector.

OUTFOX Framework

Task Formulation

Our work focuses on distinguishing LLM-generated essays from human-written essays. We assume that a training instance for detecting LLM-generated essays consists of a

[1] https://github.com/ryuryukke/OUTFOX

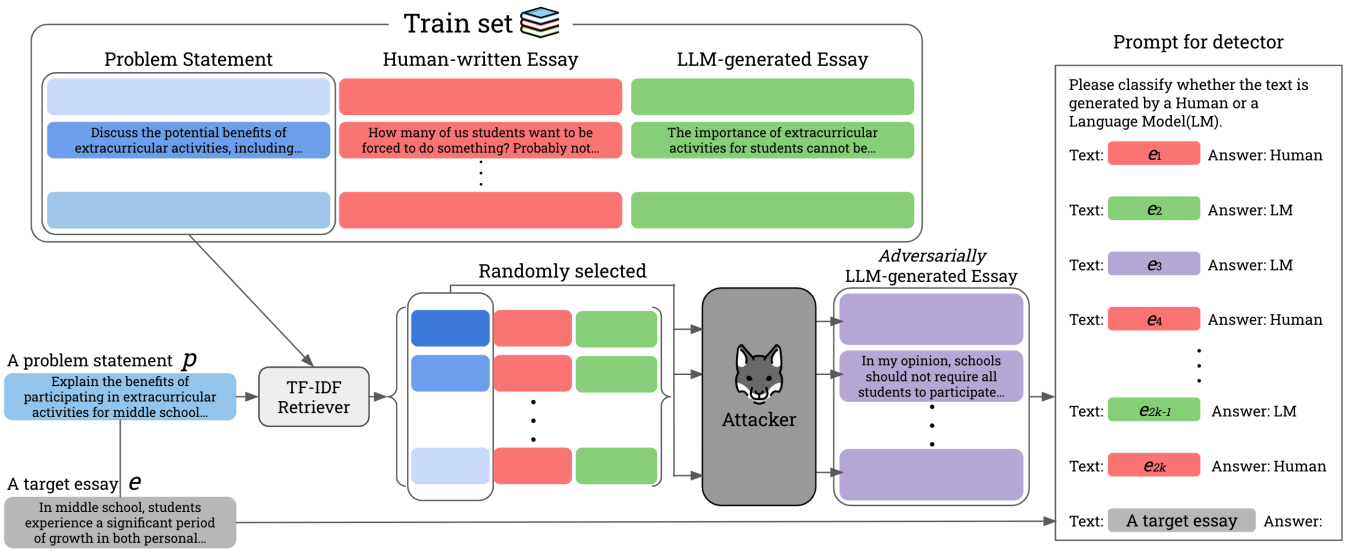

Figure 2: An illustration of our OUTFOX detector: The detector utilizes the adversarially generated essays as examples for in-context learning to learn to detect essays from our OUTFOX attacker.

triplet of an essay-problem statement, a human-written essay, and an LLM-generated essay. In our framework, the attacker adversarially generates *attacked* text, designed to fool the detector, while the detector tries to learn from examples to better distinguish LLM-generated and human-written text. Both the detector and attacker are able to *consider* each other's outputs by including examples in their input prompts for in-context learning. The detector can consider essay-label pairs that can consist of human-written, *attacked* (adversarially-generated by an attacker), and *non-attacked* (generated by an LLM, but not adversarially) essays by including them in its input prompt. On the other hand, the attacker can consider examples of non-attacked text and the detector's label predictions in its input prompt.

The OUTFOX Detector

Figure 2 illustrates the outline of our OUTFOX detector. Given a target essay e paired with a problem statement p, we retrieve the top-k problem statements that are semantically close[2] to the target problem statement p from the training set, together with human-written and LLM-generated essays associated with the retrieved problem statements. To generate attacks to be considered in detection, we randomly select j out of the k problem statements ($0 \leq j \leq k$). Consequently, our OUTFOX attacker adversarially generates an essay for each selected problem statement. In this way, we obtain k human-written, $(k-j)$ LLM-generated, and j attacked essays and construct a mixture of these essays with labels $R_{\text{det}} = \{(e_i, l_i)\}_{i=1}^{2k}$. Here, l_i is `Human` when the essay e_i is written by a human, or `LM` when the essay e_i is written by either the LLM or attacker. Tagging attacker-generated texts with `LM` label encourages our detector to learn the implicit characteristics of attacker-generated texts for detection. Finally, the detector predicts a label $\hat{l}$ for the target essay e by using the LLM with the task instruction I and R_{det} as a prompt,

$$\hat{l} = \underset{l \in \{\texttt{Human},\texttt{LM}\}}{\operatorname{argmax}} \; P\left(l \mid I, R_{\text{det}}, e\right), \tag{1}$$

$$R_{\text{det}} = \{(e_i, l_i)\}_{i=1}^{2k}. \tag{2}$$

Here, the task instruction I is "Please classify whether the text is generated by a human or a Language Model (LM)." Each example in R_{det} is converted into the text format, "Text: e_i Answer: l_i." The target essay e is converted into "Text: e. Answer: " so that the LLM predicts a label as a continuation of the prompt.

The OUTFOX Attacker

Figure 3 illustrates the workflow of our OUTFOX attacker. Unlike the paraphrasing attack, our attacker generates an essay from a problem statement so that it fools our detector. Given a problem statement p, we first retrieve the top-k problem statements p_i $(i \in \{1, \ldots, k\})$ that are semantically close[2] to the problem statement p from the training set, along with human-written and LLM-generated essays e_i associated with the retrieved problem statements. Then, our detector predicts a label $\hat{l}_i \in \{\texttt{Human}, \texttt{LM}\}$ on each retrieved LLM-generated essay e_i using in-context examples from the training set. At this time, our detector does not con-

[2]We follow the setting of Prabhumoye et al. (2022), which found that leveraging semantically close examples is effective for in-context learning. We use the vector space of all problem statements computed by Term Frequency-Inverse Document Frequency (TF-IDF) implemented in scikit-learn: https://tinyurl.com/scikitlearn-TF-IDF-vectorizer. The closeness is computed by $(1-s)$, where s presents the cosine similarity of the vectors of two problem statements.

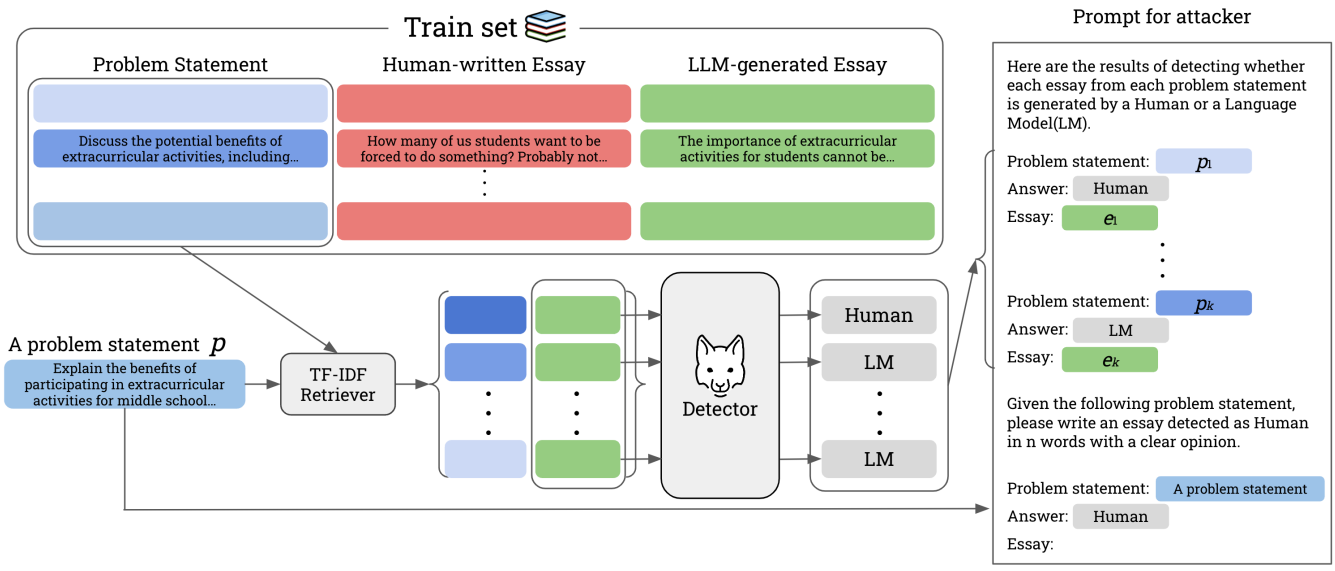

Figure 3: An illustration of our OUTFOX attacker: The attacker considers our OUTFOX detector's prediction labels as examples for in-context learning and adversarially generates essays that are harder to detect.

sider an attack.[3] Consequently, we create a mixture of the retrieved problem statements, the predicted labels, and the retrieved LLM-generated essays $R_{\text{atk}} = \{(p_i, \hat{l}_i, e_i)\}_{i=1}^{k}$. Finally, our attacker adversarially generates an essay a as a continuation for the prompt rendered from I_{d}, R_{atk}, the task instruction of essay-generation I, and the given problem statement p. We use an LLM to predict the next token $\hat{a}_t$ for the previously predicted tokens $\hat{a}_{1:t-1}$,

$$\hat{a}_t \approx \underset{a_t \in V}{\text{argmax}} P\left(a_t \mid I_{\text{d}}, R_{\text{atk}}, I, p, \hat{a}_{1:t-1}\right), \quad (3)$$

$$R_{\text{atk}} = \{(p_i, \hat{l}_i, e_i)\}_{i=1}^{k}. \quad (4)$$

Here, V is the vocabulary of the LLM. The description I_{d} is "Here are the results of detecting whether each essay from each problem statement is generated by a Human or a Language Model (LM)". Each example in R_{atk} is converted to the text format, "Problem Statement: p_i. Answer: $\hat{l}_i$. Essay: m_i." The task instruction I is "Given the following problem statement, please write an essay detected as Human in N words with a clear opinion". N is the number of words in the human-written essay paired with the given problem statement p. The problem statement p is converted into "Problem Statement: p. Answer: Human. Essay: " so that the LLM adversarially generates an essay to fool the detector as a continuation of the prompt, including explicit `Human` label.

Constructing a Dataset to Detect LLM-Generated Essays

We build a dataset for student essay writing, specifically to detect LLM-generated essays. There are already some datasets for the purpose of automatically scoring student-written essays (Hamner et al. 2012; Maggie et al. 2022), but few of them have abundant essay-problem statement pairs. To get LLM-generated essays, we need essay problem statements. We focus on the essay dataset of Maggie et al. (2022), consisting of argumentative essays written by native students from 6th to 12th grade in the U.S. Firstly, we instruct ChatGPT to generate a pseudo-problem-statement to mimic a setting where a student would produce the supplied essay. Afterward, we instruct an instruction-tuned LLM to generate an essay based on each generated problem statement. For each LLM, our dataset contains 15,400 triplets of essay problem statements, student-written essays, and LLM-generated essays. In our evaluation, we split the dataset into three parts: train/validation/test with 14400/500/500 examples, respectively. Besides the non-attacked LLM-generated essays, to evaluate each attacker in our experiments, we also build 500 attacked essays associated with problem statements in our test set for each attacker.

Experiments and Results

In our experiments, we investigate the following aspects:

- How robust is our detector, considering attacks, against attacked texts?
- Does our detector, considering attacks, consistently perform well even on non-attacked texts?
- Is our attacker stronger than the previous paraphrasing attack approach?

Overall Setup

Essay Generation Models To generate non-attacked essays, we instruct the instruction-tuned LMs: ChatGPT (gpt-

[3] Although our framework theoretically allows the detector and attacker to iteratively strengthen each other many times, we focus on only once.

Attacker	Detector	Metrics (%) ↑			
		HumanRec	MachineRec	AvgRec	F1
DIPPER	w/o Attacks	98.6	66.2	82.4	79.0
	w/ DIPPER	98.2	79.6	**88.9**	**87.8**
	w/ OUTFOX	97.8	72.4	85.1	82.9
OUTFOX	w/o Attacks	98.8	24.8	61.8	39.4
	w/ DIPPER	98.6	20.8	59.7	34.0
	w/ OUTFOX	97.2	69.6	**83.4**	**80.7**

Table 1: Comparison of the detection performances of our OUTFOX detector on attacked essays, with and without considering attacks: our OUTFOX attack and the DIPPER attack. The DIPPER paraphrases ChatGPT-generated essays for attacking. In the rows of "w/o Attacks", we show the detection performances of our detector, without considering attacks, on attacked essays by each attacker.

3.5-turbo-0301), GPT-3.5 (text-davinci-003), and FLAN-T5-XXL[4]. In each, we set the temperature parameter to 1.3.

Evaluation Metrics and Dataset Area Under Receiver Operating Characteristic curve (AUROC) can be applied only to the detectors which output real number prediction scores. Since our proposed detector outputs a binary label, we employ the F1-score on LLM-generated texts as our first metric. Our second metric for detection performance is AvgRec, following Li et al. (2023). AvgRec is the average of HumanRec and MachineRec. In our evaluation, HumanRec is the recall for detecting Human-written texts, and MachineRec is the recall for detecting LLM-generated texts. We compute a classification threshold for each baseline detector on our validation set where the Youden Index[5] (Youden 1950) is maximum in the ROC curve. Finally, we evaluate detectors with these metrics and thresholds on our test set: a mixture of 500 human-written and 500 non-attacked essays. To evaluate detectors on attacked essays, we swap only LLM-generated essays from non-attacked to attacked ones. Additionally, the threshold for each detector is fixed on both non-attacked and attacked essays.

Detection Methods Our OUTFOX detector is based on ChatGPT (gpt-3.5-turbo-0301). We set the temperature and top_p parameters to 0 in order to eliminate the randomness of our detection. Our detector takes retrieved in-context examples for an essay. In detection without considering attacks, the in-context examples are 5 human-written and 5 LLM-generated essays. In detection with considering attacks, the in-context examples are 5 human-written, 3 attacked, and 2 LLM-generated essays. Here, regardless of the essay generation models to be detected, our detector takes ChatGPT-generated essays as part of in-context examples. As a comparison with prior work, we compare our detector to the following detectors, divided into two groups: statistical outlier approaches and supervised classifiers. The first group covers Rank, LogRank, Log Probability, and DetectGPT (as explained in §). The second group contains OpenAI's RoBERTa-based GPT-2 classifiers[6] (Base, Large) and HC3 ChatGPT detector[7]. HC3 is the latest corpus targeted for detecting ChatGPT-generated texts (Guo et al. 2023). We employ default parameters for each detection method.[8]

Attacking Methods Our OUTFOX attacker is based on ChatGPT (gpt-3.5-turbo-0301). In generating attacks, we configure the temperature parameter to 1.3. Our attacker takes retrieved in-context examples for a problem statement. The number of in-context examples is 10. We compare our attacker to the paraphrasing attack by the DIPPER (as explained §). The DIPPER paraphrases the non-attacked essays for attacking. We configure both parameters for vocabulary L and content re-ordering O to 60, which are the parameters found to produce the strongest attack in Krishna et al. (2023). Other hyperparameters are set to the defaults.

Results

How Robust Is Our Detector, considering Attacks, against Attacked Texts? Table 1 presents the difference in detection performance of our OUTFOX detector with and without considering attacks on attacked essays. The attacking models include our OUTFOX attacker and the DIPPER. Throughout all attackers, our detector improves the detection performance when considering attacks. For instance, on our attacker-generated essays, our detector shows +41.3 points F1-score and +21.6 points AvgRec improvements when considering our attack. From this result, we empirically observe that our detector learns to detect essays from attackers via in-context examples. Notably, our detector, when considering our attack, shows the performance improvements on any attacked essay, while our detector considering the DIPPER improves only on the attacked essays by the DIPPER, but not our attacker. This observation suggests that our attacker is not merely paraphrasing but may generate semantically diverse essays from given problem statements. Consequently, our detector is enabled to identify texts that employ a broader spectrum of attacks.

[4]https://huggingface.co/google/flan-t5-xxl

[5]The Youden Index is the difference between True Positive Rate (TPR) and False Positive Rate (FPR). The cut-off point in the ROC curve, where the Youden Index is maximum, is the best trade-off between TPR and FPR.

[6]https://github.com/openai/gpt-2-output-dataset/tree/master/detector

[7]https://huggingface.co/Hello-SimpleAI/chatgpt-detector-roberta

[8]To adopt DetectGPT, we change only buffer_size from default to 2 in order to escape being stuck in the perturbation step.

Essay Generator	Detector	Metrics (%) ↑			
		HumanRec	MachineRec	AvgRec	F1
ChatGPT	w/o Attacks	99.0	94.0	**96.5**	**96.4**
	w/ DIPPER	99.2	87.8	93.5	93.1
	w/ OUTFOX	97.8	92.4	95.1	95.0
GPT-3.5	w/o Attacks	98.6	95.2	**96.9**	96.8
	w/ DIPPER	98.8	92.4	95.6	95.5
	w/ OUTFOX	97.6	96.2	**96.9**	**96.9**
FLAN-T5-XXL	w/o Attacks	98.8	68.2	83.5	80.5
	w/ DIPPER	99.2	72.0	**85.6**	**83.3**
	w/ OUTFOX	97.0	73.4	85.2	83.2

Table 2: Comparison of the detection performances of our OUTFOX detector on non-attacked essays, with and without considering the attacks: our OUTFOX attack and the DIPPER attack. In the rows of "w/o Attacks", we show the detection performances of our detector, without considering attacks, on non-attacked essays.

Detector	Attacker	Metrics (%) ↓			
		HumanRec	MachineRec	AvgRec	F1
RoBERTa-base	Non-attacked	93.8	92.2	93.0	92.9
	DIPPER	93.8	89.2	91.5	91.3
	OUTFOX	93.8	69.2	**81.5**	**78.9**
RoBERTa-large	Non-attacked	91.6	90.0	90.8	90.7
	DIPPER	91.6	97.0	94.3	94.4
	OUTFOX	91.6	56.2	**73.9**	**68.3**
HC3 detector	Non-attacked	79.2	70.6	74.9	73.8
	DIPPER	79.2	3.4	41.3	5.5
	OUTFOX	79.2	0.4	**39.8**	**0.7**
OUTFOX	Non-attacked	99.0	94.0	96.5	96.4
	DIPPER	98.6	66.2	82.4	79.0
	OUTFOX	98.8	24.8	**61.8**	**39.4**

Table 3: Comparison of the detection performances of the detectors on ChatGPT-generated essays, before and after being attacked: our OUTFOX attack and the DIPPER attack. In the rows of "Non-attacked", we show the detection performances of each detector on non-attacked essays.

Does Our Detector, considering Attacks, Consistently Perform Well Even on Non-attacked Texts? Table 2 shows the difference in the detection performance of our OUTFOX detector on non-attacked essays, with and without considering attacks. Our detector consistently performs well, even on non-attacked essays. The difference is minimal: an average decrease of only a -0.1 point F1-score and a -0.32 point AvgRec across all comparisons. Furthermore, in non-attacked essays by FLAN-T5-XXL, our detector performs better than without considering attacks. These results empirically show that considering attacks has little negative effects on the detection performance of our detector on the non-attacked texts.

Is Our Attacker Stronger than the Previous Paraphrasing Attack Approach? Table 3 provides the detection performance on attacked essays by different attacking approaches: our attack and the DIPPER attack. Here, our detector does not consider any attacks. Our attacker drastically degrades the detection performance of our detector and supervised classifiers the most by up to -57.0 points F1-score and -69.2 points MachineRec. In addition, our attacker has a more detrimental impact on the detection performance of all detectors than the DIPPER by up to -39.6 points F1-score. We also find that the DIPPER doesn't degrade the detection performance much and conversely improves the detection performance, especially on RoBERTa-large. This is partially because the DIPPER attack is based on paraphrasing and not designed specifically to attack detectors resulting in the opposite effect.

Comparison with Prior Work

In this section, we compare the detection performance of our OUTFOX detector, when considering our attack, and prior detectors on non-attacked essays. The prior detectors are divided into two groups: statistical outlier approaches and supervised classifiers.

Statistical Outlier Approaches Statistical outlier approaches need access to model logits for their detection. Thus, we contrast our detector with these statistical outlier approaches in detecting essays by FLAN-T5-XXL. As shown in Table 4, our detector has a far superior detection performance of 80.5 points F1-score and 83.5 points AvgRec to previous statistical approaches. We find that statistical outlier approaches tend to aggressively label Human-written essays as LLM-generated from the contrast between the low HumanRec and the high MachineRec. While the Hu-

Baseline type	Essay Generator	Detector	Metrics (%) ↑			
			HumanRec	MachineRec	AvgRec	F1
Statistical outlier methods	FLAN-T5-XXL	$\log p(x)$	2.0	97.6	49.8	66.0
		Rank	28.8	86.2	57.5	67.0
		LogRank	12.0	90.6	51.3	65.0
		Entropy	39.4	80.4	59.9	66.7
		DetectGPT	29.8	76.2	53.0	61.9
		OUTFOX	97.0	73.4	**85.2**	**83.2**
Supervised classifiers	ChatGPT	RoBERTa-base	93.8	92.2	93.0	92.9
		RoBERTa-large	91.6	90.0	90.8	90.7
		HC3 detector	79.2	70.6	74.9	73.8
		OUTFOX	97.8	92.4	**95.1**	**95.0**
	GPT-3.5	RoBERTa-base	93.8	92.0	92.9	92.8
		RoBERTa-large	92.6	92.0	92.3	92.3
		HC3 detector	79.2	85.0	82.1	82.6
		OUTFOX	97.6	96.2	**96.9**	**96.9**

Table 4: Comparison of the detection performances of our OUTFOX detector and prior approaches on non-attacked essays. Prior approaches include statistical outlier detectors and supervised classifiers. We compare our detector with statistical outlier detectors on the essays by FLAN-T5-XXL and supervised classifiers on the essays by ChatGPT and GPT-3.5.

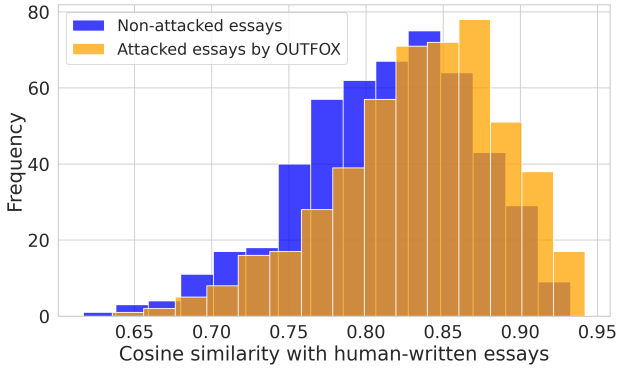

Figure 4: Cosine similarity distributions of non-attacked essays and our OUTFOX attacker-generated essays with human-written essays, respectively.

manRec of our detector is high: 98.8 points, partially because our detector is based on ChatGPT, which is trained with human feedback, thus avoiding aggressive detection.

Supervised Classifiers We compare our detector and supervised classifiers in detecting essays generated by each ChatGPT and GPT-3.5. Table 4 shows that our detector has better detection performance on both essays by ChatGPT and GPT-3.5 than supervised classifiers. We empirically find that ChatGPT has the few-shot ability to detect LLM-generated texts via labeled in-context examples without any parameter updates.

In summary, our OUTFOX detector shows the state-of-the-art detection performance on non-attacked essays.

The Semantic Similarity of Our OUTFOX Attacks with Human-Written Essays

To explore why our attacker substantially degrades the detectors' performance, we investigate the difference between the semantic similarity of non-attacked essays and our attacker-generated essays with human-written essays. We focus on our test set, consisting of 500 human-written, 500 ChatGPT-generated, and 500 attacked essays generated by our attacker. For determining semantic similarity between two essays, we employ a pre-trained BERT model[9] to each essay and compute a cosine similarity of the resulting embeddings. Figure 4 presents cosine similarity distributions of non-attacked and our attacker-generated essays with human-written essays. A rightward shift implies that our attacker-generated essays are more semantically similar to human-written essays than non-attacked essays. From the shift, we empirically find that our OUTFOX attacker can generate essays that are more semantically similar to human-written essays than the non-attacking normal LLM, leading to difficulty in detecting attacker-generated essays.

Conclusion

We proposed OUTFOX, a framework that improves the robustness of the detector against attacks by allowing both the detector and the attacker to consider each other's outputs as examples for in-context learning. The experiments in the domain of student essays demonstrate that 1) Our detector can learn to detect essays from attackers via in-context examples and 2) Notably, considering attacks of our detector has little negative effect on the detection of non-attacked texts. and 3) Our attacker, which is designed specifically to deceive the detector, can evade current LLM-generated text detectors more effectively than the previous paraphrasing attack. Furthermore, our analysis reveals that our attacker can generate an essay that is semantically closer to a human-written essay than a non-attacked essay, leading to success in effective attacking. In future work, we will apply our framework to other domains, such as fake news generation and academic paper writing.

[9]https://huggingface.co/bert-large-cased

Acknowledgements

These research results were obtained from the commissioned research (No.22501) by National Institute of Information and Communications Technology (NICT), Japan.

References

Badaskar, S.; Agarwal, S.; and Arora, S. 2008. Identifying Real or Fake Articles: Towards better Language Modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Bakhtin, A.; Gross, S.; Ott, M.; Deng, Y.; Ranzato, M.; and Szlam, A. 2019. Real or Fake? Learning to Discriminate Machine from Human Generated Text. arXiv:1906.03351.

Beresneva, D. 2016. Computer-generated text detection using machine learning: A systematic review. In *21st International Conference on Applications of Natural Language to Information Systems, NLDB*, 421–426. Springer.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Crothers, E.; Japkowicz, N.; and Viktor, H. 2023. Machine Generated Text: A Comprehensive Survey of Threat Models and Detection Methods. arXiv:2210.07321.

Google. 2023. Google AI updates: Bard and new AI features in Search. Accessed: 2023-05-10.

Guo, B.; Zhang, X.; Wang, Z.; Jiang, M.; Nie, J.; Ding, Y.; Yue, J.; and Wu, Y. 2023. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. arXiv:2301.07597.

Hamner, B.; lynnvandev, J. M.; Shermis, M.; and Ark, T. V. 2012. The Hewlett Foundation: Automated Essay Scoring. Accessed: 2023-05-10.

Hu, X.; Chen, P.-Y.; and Ho, T.-Y. 2023. RADAR: Robust AI-Text Detection via Adversarial Learning. arXiv:2307.03838.

Ibrahim, H.; Liu, F.; Asim, R.; Battu, B.; Benabderrahmane, S.; Alhafni, B.; Adnan, W.; Alhanai, T.; AlShebli, B.; Baghdadi, R.; Bélanger, J. J.; Beretta, E.; Celik, K.; Chaqfeh, M.; Daqaq, M. F.; Bernoussi, Z. E.; Fougnie, D.; de Soto, B. G.; Gandolfi, A.; Gyorgy, A.; Habash, N.; Harris, J. A.; Kaufman, A.; Kirousis, L.; Kocak, K.; Lee, K.; Lee, S. S.; Malik, S.; Maniatakos, M.; Melcher, D.; Mourad, A.; Park, M.; Rasras, M.; Reuben, A.; Zantout, D.; Gleason, N. W.; Makovi, K.; Rahwan, T.; and Zaki, Y. 2023. Perception, performance, and detectability of conversational artificial intelligence across 32 university courses. arXiv:2305.13934.

Ippolito, D.; Duckworth, D.; Callison-Burch, C.; and Eck, D. 2020. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1808–1822. Online: Association for Computational Linguistics.

Kirchenbauer, J.; Geiping, J.; Wen, Y.; Katz, J.; Miers, I.; and Goldstein, T. 2023. A Watermark for Large Language Models. arXiv:2301.10226.

Krishna, K.; Song, Y.; Karpinska, M.; Wieting, J.; and Iyyer, M. 2023. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. arXiv:2303.13408.

Lavergne, T.; Urvoy, T.; and Yvon, F. 2008. Detecting Fake Content with Relative Entropy Scoring. In *Proceedings of the ECAI'08 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*, CEUR Workshop Proceedings.

Li, Y.; Li, Q.; Cui, L.; Bi, W.; Wang, L.; Yang, L.; Shi, S.; and Zhang, Y. 2023. Deepfake Text Detection in the Wild. arXiv:2305.13242.

Liu, Y.; Zhang, Z.; Zhang, W.; Yue, S.; Zhao, X.; Cheng, X.; Zhang, Y.; and Hu, H. 2023. ArguGPT: evaluating, understanding and identifying argumentative essays generated by GPT models. arXiv:2304.07666.

Maggie, A. F.; Benner, M.; Rambis, N.; Baffour, P.; Holbrook, R.; and ulrichboser, S. C. 2022. Feedback Prize - Predicting Effective Arguments. Accessed: 2023-05-10.

Mitchell, E.; Lee, Y.; Khazatsky, A.; Manning, C. D.; and Finn, C. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. arXiv:2301.11305.

OpenAI. 2023a. AI Text Classifier. Accessed: 2023-05-10.

OpenAI. 2023b. Educator considerations for ChatGPT. Accessed: 2023-05-10.

OpenAI. 2023c. Introducing ChatGPT. Accessed: 2023-05-10.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155.

Prabhumoye, S.; Kocielnik, R.; Shoeybi, M.; Anandkumar, A.; and Catanzaro, B. 2022. Few-shot Instruction Prompts for Pretrained Language Models to Detect Social Biases. arXiv:2112.07868.

Rodriguez, J.; Hay, T.; Gros, D.; Shamsi, Z.; and Srinivasan, R. 2022. Cross-Domain Detection of GPT-2-Generated Technical Text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1213–1233. Seattle, United States: Association for Computational Linguistics.

Sadasivan, V. S.; Kumar, A.; Balasubramanian, S.; Wang, W.; and Feizi, S. 2023. Can AI-Generated Text be Reliably Detected? arXiv:2303.11156.

Sanh, V.; Webson, A.; Raffel, C.; Bach, S.; Sutawika, L.; Alyafeai, Z.; Chaffin, A.; Stiegler, A.; Raja, A.; Dey, M.; Bari, M. S.; Xu, C.; Thakker, U.; Sharma, S. S.; Szczechla, E.; Kim, T.; Chhablani, G.; Nayak, N.; Datta, D.; Chang, J.; Jiang, M. T.-J.; Wang, H.; Manica, M.; Shen, S.; Yong, Z. X.; Pandey, H.; Bawden, R.; Wang, T.; Neeraj, T.; Rozen,

J.; Sharma, A.; Santilli, A.; Fevry, T.; Fries, J. A.; Teehan, R.; Scao, T. L.; Biderman, S.; Gao, L.; Wolf, T.; and Rush, A. M. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*.

Solaiman, I.; Brundage, M.; Clark, J.; Askell, A.; Herbert-Voss, A.; Wu, J.; Radford, A.; Krueger, G.; Kim, J. W.; Kreps, S.; McCain, M.; Newhouse, A.; Blazakis, J.; McGuffie, K.; and Wang, J. 2019. Release Strategies and the Social Impacts of Language Models. arXiv:1908.09203.

Tang, R.; Chuang, Y.-N.; and Hu, X. 2023. The Science of Detecting LLM-Generated Texts. arXiv:2303.07205.

Uchendu, A.; Le, T.; Shu, K.; and Lee, D. 2020. Authorship Attribution for Neural Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8384–8395. Online: Association for Computational Linguistics.

Vasilatos, C.; Alam, M.; Rahwan, T.; Zaki, Y.; and Maniatakos, M. 2023. HowkGPT: Investigating the Detection of ChatGPT-generated University Student Homework through Context-Aware Perplexity Analysis. arXiv:2305.18226.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682.

Youden, W. J. 1950. An index for rating diagnostic tests. *Cancer*, 32–35.