Mitigating Paraphrase Attacks on Machine-Text Detectors via Paraphrase Inversion

Rafael Rivera Soto[1,2], Barry Chen[1], and Nicholas Andrews[2]

[1]Lawrence Livermore National Laboratory
[2]Johns Hopkins University
[1]rafaelriverasoto@jhu.edu, chen52@llnl.gov, noa@jhu.edu

Abstract

High-quality paraphrases are easy to produce using instruction-tuned language models or specialized paraphrasing models. Although this capability has a variety of benign applications, *paraphrasing attacks*—paraphrases applied to machine-generated texts—are known to significantly degrade the performance of machine-text detectors. This motivates us to consider the novel problem of paraphrase inversion, where, given paraphrased text, the objective is to recover an approximation of the original text. The closer the approximation is to the original text, the better machine-text detectors will perform. We propose an approach which frames the problem as translation from paraphrased text back to the original text, which requires examples of texts and corresponding paraphrases to train the *inversion* model. Fortunately, such training data can easily be generated, given a corpus of original texts and one or more paraphrasing models. We find that language models such as GPT-4 and Llama-3 exhibit biases when paraphrasing which an inversion model can learn with a modest amount of data. Perhaps surprisingly, we also find that such models generalize well, including to paraphrase models unseen at training time. Finally, we show that when combined with a paraphrased-text detector, our inversion models provide an effective defense against paraphrasing attacks, and overall our approach yields an average improvement of +22% AUROC across seven machine-text detectors and three different domains.

1 Introduction

Recent developments in the capabilities of large language models (LLMs) such as GPT-4 (OpenAI et al., 2024) have resulted in their widespread use by a variety of users. Although most users act responsibly, there is growing concern about abuses of LLMs, such as for plagiarism, spam, or spreading misinformation (Weidinger et al., 2022; Hazell, 2023). To minimize the abuse of these systems, several machine-text detection systems have been proposed, including Binoculars (Hans et al., 2024), FastDetectGPT (Bao et al., 2024), and watermarking-based algorithms (Kirchenbauer et al., 2023; Kuditipudi et al., 2024). However, these systems often fail to detect text that has been paraphrased by another model (Krishna et al., 2020; Sadasivan et al., 2025), leaving a critical gap in current detection systems.

To tackle this issue, a recent study has proposed jointly training a paraphraser and a machine-text detector with an adversarial objective: the paraphraser generates text to evade detection, while the detector identifies paraphrased text (Hu et al., 2023). Another study has proposed that LLM API providers cache their generations, enabling retrieval over a semantic space, where candidates with high similarity to previous generations are marked as paraphrases (Krishna et al., 2023). Unfortunately, both approaches lack generality, as they depend on training a specialized detector, or having access to all model generations. A more desirable defense would be *detector agnostic*, improving the performance of any detector.

Ideally, if the original tokens of a paraphrased text could be recovered, machine-text detectors would perform well, eliminating the need for any specialized solutions. Therefore, we propose the novel task of *paraphrase inversion*, where the objective is to recover the original text from a paraphrased one. This approach has the added benefit of being detector agnostic. Given the space of possible paraphrases and the stochastic sampling procedures commonly used, inverting paraphrased text is challenging. Nonetheless, there is evidence that LLMs exhibit consistent biases even when the instruction implicitly or explicitly requests diversity in the responses (Zhang et al., 2024b; Wu et al., 2025).

Even if paraphrase inversion is possible, we must know *when* to apply it, making paraphrase detec-

Findings of the Association for Computational Linguistics: ACL 2025, pages 4421–4433
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

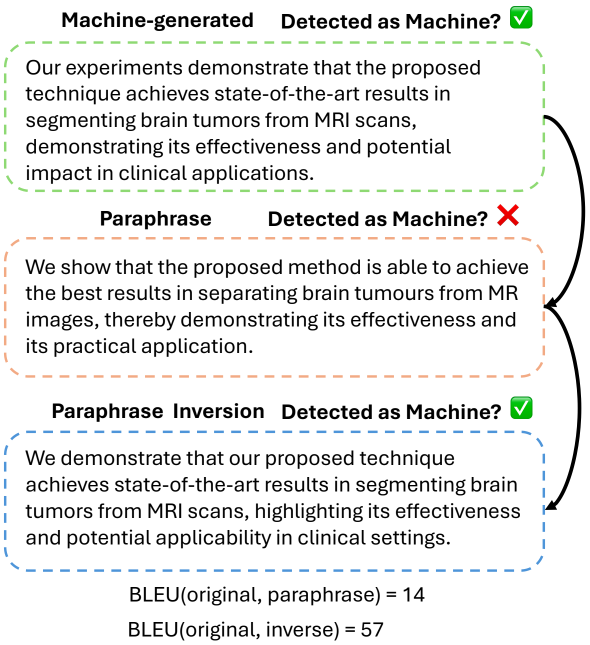

Figure 1: Paraphrasing defeats machine-text detection system. Our proposed defense (§3) consists of two steps: (1) detecting whether text is a paraphrase, and (2) if so, (2) inverting the paraphrase back to the original text. This pipeline improves the AUROC of 7 machine-text detectors across three domains by an average of +22% AUROC (Table 1).

tion a necessary step. Detecting text as having undergone LLM paraphrasing differs from detecting it as machine-generated, as the original text may have been human-written, in which case large portions of the original document may be copies of the human-written original. In cases where the original text is human-written, a machine-text detector should classify it as such, for example in cases where an LLM is used as a writing assistant.

To address these concerns, we propose *paraphrase detection* and *paraphrase inversion* as a pipeline to improve the performance of any machine-text detector in scenarios where texts may have been paraphrased (Figure 1). Our main contributions are as follows:

- We introduce the task of paraphrase inversion (§3), where the goal is to recover the original text from a paraphrased one. We formalize the task and provide a comprehensive analysis of its challenges. We find that inverting human-written text is significantly harder than inverting paraphrases of machine-generated text, which is to be expected given that human-written text exhibits higher entropy under LLM distributions (Gehrmann et al., 2019).
- We explore two paraphrase detection schemes: (1) a simple neural classifier trained to detect paraphrased text and (2) an approach that leverages our paraphrase inversion model directly without requiring an additional model (§3.3).
- We combine paraphrase detection and paraphrase inversion into a single pipeline that improves the detection rate of seven machine-text detectors across three domains (§5.2) by an average of +22% AUROC.

Reproducibility The dataset, method implementations, model checkpoints, and experimental scripts, will be released along with the paper.[1]

2 Related Work

Paraphrasing A number of paraphrase corpora have been released over the years which has enabled the development of paraphrase detection and generation models (Dolan and Brockett, 2005; Ganitkevitch et al., 2013; Wieting and Gimpel, 2018; Zhang et al., 2019; Krishna et al., 2020). Paraphrases have been shown to degrade the performance of machine-text detectors, including those based upon watermarking (Krishna et al., 2023; Sadasivan et al., 2025). In response to this, several defenses have been proposed, including jointly training a paraphraser and a detector in an adversarial setting (Hu et al., 2023), building specialized detectors for both the paraphrasing model and the language model (Soto et al., 2024), and retrieval over a database of semantically similar generations produced by the model in the past (Krishna et al., 2023). Paraphrases have also been shown to be an effective attack against authorship verification systems (Potthast et al., 2016; Wang et al., 2023), allowing bad actors to conceal their identity. To our knowledge, our approach is the first attempt at inverting the paraphrases, both in general and in the context of defending against paraphrasing attacks on machine-text detection.

Embedding inversion Several lines of work, both in computer vision (Mahendran and Vedaldi, 2015; Teterwak et al., 2021; Dosovitskiy and Brox, 2016) and natural language processing (Song and Raghunathan, 2020; Li et al., 2023; Morris et al., 2023) have explored whether embeddings can be inverted back to their inputs. Prior work has shown

[1] Code for all experiments available `https://github.com/rrivera1849/inversion`

that it is possible to recover 92% of 32-token text inputs given semantic embeddings (Morris et al., 2024). Moreover, even when the text is isn't recovered with high-fidelity, sensitive attributes such as the authorship are recoverable (Song and Raghunathan, 2020). In computer vision, even when an inversion model is applied to an adversarially robust classifier, enough local and global detail remains, making the inversion confusable with the original image, highlighting the difficulty of safeguarding sensitive attributes (Teterwak et al., 2021). Inverting embeddings is significantly easier than inverting paraphrases, as embeddings encode rich features of their inputs in *continuous* latent-space, in contrast to the *discrete* space of paraphrased tokens.

Language model inversion (Morris et al., 2024) The objective here is to recover the prompt that generated a particular output. Language model inversion techniques such as `logit2text` (Morris et al., 2024) require knowledge of the LLM that generated the output *and* access to the next-token probability distribution, making it difficult to apply in practice. Another approach more closely related to ours is `output2prompt` (Zhang et al., 2024a), which trains an encoder-decoder architecture to generate the prompt given *multiple* outputs. However, `output2prompt` requires upwards of 16 outputs per prompt to successfully match the performance of `logit2text`, and only handles prompts up to 64 tokens long. In contrast to these methods, we focus exclusively on inverting LLM-generated paraphrases given a *single* example *cleaned* of all obvious generation artifacts such as "`note: I changed...`", thereby removing all telltale signs of what the original text might've been[2]. Therefore, the paraphrase inversion problem considered in this paper is more challenging than related problems posed in prior work.

3 Methods

3.1 Overview

Given a text sample y_i, we first detect whether it is a paraphrase using one of our detection schemes. If it is classified as a paraphrase, we apply our paraphrase inversion model to recover the original text $\hat{x} \sim p(. \mid y_i)$. This sample is then run through a machine-text detector.

[2]More details can be found in Appendix G

Paraphrase inversion The task of reconstructing the original source text given paraphrased text. The difficulty of this task hinges in large part on assumptions regarding the paraphrasing model. We assume access to one or more paraphrasing models from which we can generate new paraphrases $\{y_i\}_{i=1}^N$ given a corpus of N source documents $\{x_i\}_{i=1}^N$. While access to the paraphrasing models in principle affords the possibility of producing an arbitrary amount of training data, in practice the paraphraser may be associated with non-trivial inference costs (e.g., GPT-4). Moreover, even if the paraphrasing model is known, the decoding parameters such as temperature may not be.[3] Therefore, a key question is whether paraphrase inversion models generalize to *unseen* paraphrasers, which we consider in §6.3.

Paraphrase detection The goal is to identify whether a given text is the output of an LLM paraphraser, *regardless of whether original text was human-written or machine-generated*. Paraphrase detection is crucial for machine-text detection in the wild, where determining *when* to apply a paraphrase inversion model is necessary. We emphasize that detecting text as a paraphrase is not the same as identifying text as machine-generated, as the original text may have been human-written. In cases where the original text is human-written, a machine-text detector should classify it as such. This highlights the need of applying a paraphrase inversion model to ensure correct detection. However, such a pipeline raises the risk of propagation of errors, and we should therefore carefully consider the cost of such errors.

1. A false positive occurs when a non-paraphrased text is misidentified as paraphrased. To minimize the impact of such errors, a robust paraphrase inversion model should make *minimal changes* to the text in such cases. We find that our models make significantly fewer changes to non-paraphrased documents (§3.3), and that this can in fact be used as a way to distinguish between paraphrased and non-paraphrased text.
2. A false negative occurs when a paraphrased text is missed by the detector and we fail to apply the inversion model. In this case, the machine-text detector is applied to the unmodified paraphrased text, which if the original text was machine-generated, is likely to result in

[3]We investigate the impact of varying sampling the temperature during training and inference in Appendix C.

falsely predicting that it is human written.

Given the above considerations, the paraphrase text detector should aim for high recall at the cost of potentially lower precision.

3.2 End-to-end paraphrase inversion

Training objective The inversion models considered in this paper are fine-tuned using the standard supervised text-to-text objective, fitting an autoregressive conditional language model $p_\theta(y_i \mid x_i)$ on the basis of observed pairs of texts and their paraphrases (x_i, y_i). Our datasets are described in §4.1. We parameterize all our inversion models using Mistral-7B[4], training it with the hyper-parameters shown in Appendix F. We use teacher forcing during training, conditioning on the the true observed tokens.

Inference However good the paraphrasing model, there may be considerable uncertainty in the distribution over the original text. Therefore we sample several inversions and use a scoring function to select a single sample which scores highest.

Choice of score A number of criteria could be optimized to help select a single inversion likely to be close to the original text. For example, inversions should retain the meaning of the paraphrased text, and so the score could include a measure of semantic similarity. Furthermore, inspired by the finding of Soto et al. (2024) that machine-generated text is *stylistically* distinct from human-written text, we posit that the inversion should be stylistically distinct from the paraphrased text, as this would indicate a return to the original machine or human style. In preliminary experiments, we found that the paraphrasing model consistently preserved meaning in generated samples, and so to avoid introducing additional hyper-parameters and computational expense, we focus on stylistic distinctness. Specifically, we compute a stylistic embedding of the samples and original text to compute a stylistic distance for each candidate inversion, and select the inversion which is *furthest*—the most stylistically distinct.

3.3 Detecting Paraphrases

Neural paraphrase detector In the simplest case, we train a paraphrase detector $d_\phi(. \mid y_i)$ using the standard binary-cross-entropy classification loss. In addition to the standard loss, we optimize the model for a paraphrased token prediction task, where the goal is to determine whether each token in a document is copied from the original text or paraphrased. We include this loss to help the model capture the biases that paraphrasers introduce when rewording text. We optimize the binary-cross-entropy for each token, corresponding to independent classification decisions. Our model is initialized from `RoBERTa-large`[5] (Liu et al., 2019), with a multi-layer-perceptron (MLP) head that predicts whether each token was copied from the original text or paraphrased.

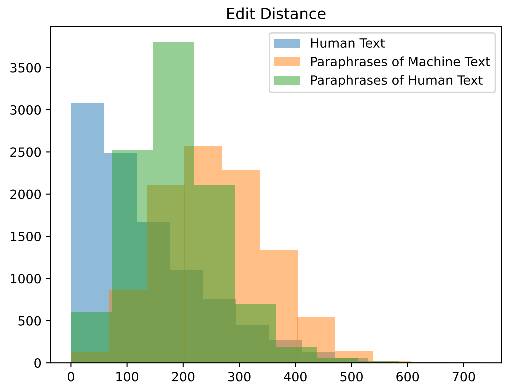

Figure 2: Edit distances between the original text and its inversion when the machine-paraphrase inversion model is applied to human-text and paraphrases of human- or machine-text. The inversion model edits human-written significantly less.

Edit-based paraphrase detector Rather than training a neural classifier, we determine whether a sample y_i is a paraphrase based on how many edits our paraphrase inversion model makes. Intuitively, if the paraphrase inversion model captures LLM paraphrasing biases, it should make *fewer* edits when "inverting" a human-written text than when inverting a paraphrase. Indeed, we find that this is the case in Figure 2. This observation motivates the following paraphrase detection scheme. Given two Gaussian distributions g_h and g_m, where g_h is fit on edit distances of human-text inversions and their originals and g_m on those from paraphrases of human- and machine-text and their inversions, we detect whether a sample y_i is a paraphrase by calculating whether y_i is more probable under g_m than g_h. This is equivalent to applying a likelihood-ratio test with a threshold of 1. In practice, because we have N inversions per sample, we take the majority

[4] `mistralai/Mistral-7B-Instruct-v0.3`

[5] `FacebookAI/roberta-large`

vote of all such predictions.

4 Experimental Procedure

4.1 Datasets

We evaluate our approach on three domains: `Reddit`, `ArXiv`, and `MovieReviews`. We use `Reddit` specifically to test the *feasibility* of paraphrase inversion, while all three domains are used to evaluate our pipeline for defending machine-text detectors against paraphrase attacks. The validation sets of each domain are used to train our edit-based paraphrase detector introduced in §3.3, while the training sets are used to train our paraphrase inversion models and our neural paraphrase detector. The `ArXiv` and `MovieReviews` datasets are subsampled from the RAID (Dugan et al., 2024) dataset, a machine-text detection benchmark which contains paraphrases of machine-text using DIPPER (Krishna et al., 2023). We refer to these two datasets as `RAID-ArXiv` and `RAID-MovieReviews`. The details of how RAID was subsampled can be found in Appendix B. Here, we discuss how we generate human-text paraphrases and machine-text paraphrases for the `Reddit` domain, as well as the construction of the `Reddit` machine-detection dataset.

Human-text paraphrases We use the Reddit Million User Dataset (MUD), which contains comments from over 1 million Reddit users over a wide variety of topics (Khan et al., 2021). We subsample the dataset according to the procedure in Appendix A. Once subsampled, we generate the paraphrases of human-text by prompting Mistral-7B[6] (Jiang et al., 2023), Phi-3B[7] (Abdin et al., 2024), and Llama-3.1-8B[8] (Dubey et al., 2024). We clean all obvious LLM-generated artifacts such as `This rephrased passage condenses`, `note: I changed...`, and ensure that all paraphrases have a semantic similarity of at least 0.7 under SBERT[9] (Reimers and Gurevych, 2019).

Machine-text paraphrases To generate paraphrases of *machine-text*, we first prompt one of the three LLM at random to produce a response to each human-written comment, then we follow the same paraphrasing procedure described above.

[6]`mistralai/Mistral-7B-Instruct-v0.3`
[7]`microsoft/Phi-3-mini-4k-instruct`
[8]`meta-llama/Meta-Llama-3-8B-Instruct`
[9]`sentence-transformers/all-mpnet-base-v2`

Machine-text detection We combine the test set of both our human-text paraphrase and machine-text paraphrase datasets to create a new set composed of 500 samples in each category: human text, paraphrases of human text, and paraphrases of machine text.

4.2 Metrics

To measure how well the inverted text recovers the true tokens, we make use of BLEU (Papineni et al., 2002), a measure of n-gram overlap. Recovering the original tokens may be difficult, if not impossible. As such, we posit that the inverted text should be close both in style and semantics to the original. We measure the stylistic similarity by embedding the inversion and the original using `LUAR` (Rivera-Soto et al., 2021)[10], a model that captures the stylistic features of text; we report the stylistic similarity as the cosine similarity between the embeddings. For semantic similarity, we use `SBERT` (Reimers and Gurevych, 2019) to embed the texts and report the cosine similarity between them. To test the performance of the machine-text detectors, we report the area under the curve (`AUC`) of the receiver operating curve (`ROC`), here denoted as `AUROC`. The `ROC` curve captures the trade-off between false positive and true positive rates across all decision thresholds; the `AUROC` summarizes this performance as a single scalar. To measure the performance of the paraphrase detectors, we use the `F1` score which measures the harmonic mean between the precision and recall achieved by the detector.

4.3 Baselines

For comparison, we prompt GPT-4 to invert the paraphrases. We report the prompts used in §D.2. Additionally, we compare our inversion model to `output2prompt` (Zhang et al., 2024a), training it on the same dataset. For machine-text detection, we avail of many popular detectors. We use Rank (Gehrmann et al., 2019), LogRank (Solaiman et al., 2019), Entropy (Ippolito et al., 2020), OpenAI's detector (Solaiman et al., 2019), RADAR (Hu et al., 2023), FastDetectGPT (Bao et al., 2024), and Binoculars (Hans et al., 2024).

5 Main Results

This section present results for our motivating application of defending against paraphrasing attacks

[10]`rrivera1849/LUAR-CRUD`

Detector	AUROC		
	Baseline	Inversion+Edit-based	Inversion+Neural
Reddit			
OpenAI (2019)	0.56	0.77	**0.79**
Rank (2019)	0.56	0.66	**0.68**
LogRank (2019)	0.58	0.74	**0.77**
Entropy (2020)	0.51	**0.59**	**0.59**
RADAR (2023)	0.62	0.66	**0.70**
FastDetectGPT (2024)	0.66	0.80	**0.84**
Binoculars (2024)	0.77	0.84	**0.89**
RAID-ArXiv			
OpenAI (2019)	**0.81**	0.79	0.77
Rank (2019)	0.71	0.69	**0.79**
LogRank (2019)	0.75	0.72	**0.91**
Entropy (2020)	0.39	0.42	**0.62**
RADAR (2023)	**0.99**	0.98	**0.99**
FastDetectGPT (2024)	0.83	0.78	**0.91**
Binoculars (2024)	0.92	0.86	**0.98**
RAID-MovieReviews			
OpenAI (2019)	0.82	0.77	**0.83**
Rank (2019)	0.60	0.76	**0.84**
LogRank (2019)	0.66	0.84	**0.91**
Entropy (2020)	0.39	0.63	**0.71**
RADAR (2023)	0.92	0.92	**0.95**
FastDetectGPT (2024)	0.74	0.80	**0.89**
Binoculars (2024)	0.91	0.92	**0.96**

Table 1: Machine-text detection performance on a dataset of human-text, paraphrases of human-text, and paraphrases of machine-text. Applying our inversion model to all samples detected as paraphrases using our paraphrase detection schemes (§3.3), we observe significant improvements in detection performance.

Dataset	Edit-based	Neural
Reddit	0.79	**0.94**
RAID-ArXiv	0.52	**0.67**
RAID-Reviews	**0.79**	0.72

Table 2: F1 scores for the proposed paraphrased detection schemes (§3.3).

for machine-text detection. Next, in §6, we perform further analysis of individual components of our approach, including the feasibility of paraphrase inversion as a stand-alone task, considering both inversions of paraphrased machine-generated (§6.1) and inversions paraphrased human-written documents (§6.2).

5.1 Paraphrase detection

We evaluate the proposed paraphrased detection schemes described in §3.3. We train the methods in all three domains, and report results in Table 2. We find that the neural detector outperforms the edit-based detector across two out of three of the domains. Moreover, the edit-based detector performs poorly in RAID-ArXiv, the most challenging domain, which in turn harms the performance of machine-text detectors in this setting (§5.2).

5.2 Machine-Text Detection

We consider the scenario where human- or machine-text may have been paraphrased by an LLM. In this scenario, it would be desirable to label paraphrases of human-text as human-written and paraphrases of machine-text as machine-generated. We train and evaluate our defense pipeline on all three domains separately. We run our paraphrase detection schemes on the held-out test set, inverting each sample detected as a paraphrase 100 times, and picking the inversion that is the *farthest* away from the input-text in `LUAR` space, ensuring that the style is dissimilar from paraphrasing style. We report the `AUROC` of 7 popular machine-text detectors in Table 1, and make the following observations: (1) Our defense, with the neural paraphrase detector improves the performance of 7 machine-text detectors across 3 domains. The only exception is OpenAI's detector on the `RAID-ArXiv` dataset. (2) RADAR, a detector designed to be robust against paraphrase attacks, also benefits. Indeed, in the worst case, RADAR's performance remains unchanged (`RADAR-ArXiv`), but in other domains, we observe notable improvements. This highlights that our defense can be combined with other existing defenses. (3) The

Method	Type	Machine-written Text			Human-written Text		
		Style (↑)	Meaning (↑)	BLEU (↑)	Style (↑)	Meaning (↑)	BLEU (↑)
Paraphrases	-	0.80	0.88	0.17	0.51	0.82	0.08
Baselines							
GPT-4	Single	0.80	0.85	0.20	0.50	0.80	0.07
	Max	0.86	0.90	0.33	0.56	0.84	0.11
	Mean	0.80	0.87	0.21	0.50	0.80	0.07
out2prompt	Single	0.48	0.17	0.00	0.39	0.10	0.00
	Max	0.71	0.40	0.04	0.53	0.32	0.02
	Mean	0.48	0.17	0.00	0.39	0.09	0.00
Ours							
Inversion	Single	0.84	0.90	0.34	0.54	0.81	0.13
	Max	**0.91**	**0.95**	**0.51**	**0.70**	**0.90**	**0.25**
	Mean	0.84	0.90	0.35	0.54	0.81	0.12

Table 3: Results of inverting *paraphrases of machine-written text* (left three columns) and *paraphrases of human-written text* (right three columns). We generate 100 inversions per sample and report the metrics achieved by a single inversion, by the best inversion (max), and the average across all inversions. Our proposed inversion model outperforms all baselines.

Detector	AUROC	
	Baseline	Inversion
Train - RAID-MovieReviews, Eval - RAID-ArXiv		
OpenAI (2019)	0.81	**0.84**
Rank (2019)	0.71	**0.83**
LogRank (2019)	0.75	**0.89**
Entropy (2020)	0.39	**0.68**
RADAR (2023)	**0.99**	**0.99**
FastDetectGPT (2024)	0.83	**0.90**
Binoculars (2024)	0.92	**0.96**
Train - RAID-ArXiv, Eval - RAID-MovieReviews		
OpenAI (2019)	**0.82**	**0.82**
Rank (2019)	0.60	**0.83**
LogRank (2019)	0.66	**0.90**
Entropy (2020)	0.39	**0.68**
RADAR (2023)	0.92	**0.94**
FastDetectGPT (2024)	0.74	**0.87**
Binoculars (2024)	0.91	**0.95**

Table 4: Machine-text detection performance on a dataset of human-text, paraphrases of human-text, and paraphrases of machine-text. We find that when our pipeline generalizes even when trained on one domain, and evaluated on another (e.g. `RAID-ArXiv` → `RAID-MovieReviews`).

edit-based paraphrase detector is not robust across all domains. Although the edit-based paraphrase detector improves performance on the `Reddit` and `RAID-MovieReviews` datasets, it reduces performance on `RAID-ArXiv`. This decline is due to the many mis-classifications in that domain. However, overall we observe an average improvement of +22% `AUROC` averaged across all detectors and domains.

5.3 Generalizing across domains

Do the paraphrase detection and paraphrase inversion models generalize from one dataset to another? We apply the pipeline using the neural paraphrase detector and inversion model trained on `RAID-ArXiv` to `RAID-MovieReviews`, and vice versa, showing our results in Table 4. We find that our pipeline improves results across all detectors even under these conditions, suggesting that paraphrasers exhibit similar biases regardless of what domain they're applied to.

6 Further Analysis

6.1 Inverting paraphrases of machine-generated text

In this section, we explore the extent to which paraphrases of *machine-generated* text can be inverted to their original tokens. We expect this task to be easier than inverting paraphrases of *human-written* text, as human-written tokens exhibit high entropy under LLM distributions (Gehrmann et al., 2019). We train and evaluate all models on Reddit, generating 100 inversions per sample on the held-out test set and report metrics in Table 3. We observe that our model recovers significant portions of the original text, with the best-scoring inversions achieving an average BLEU score of 51, with semantic and stylistic similarities of 0.95 and 0.91, respectively.

6.2 Inverting paraphrases of human-written text

We now turn to the more difficult problem of inverting paraphrases of *human-written* text. We train and evaluate all models on Reddit, generating 100 inversions per sample on the held-out test set and report metrics in Table 3. We highlight some key observations: (1) Inverting paraphrases

Method	Type	Style Sim. (↑)	Semantic Sim. (↑)	BLEU (↑)
Paraphrases	-	0.61	0.90	0.21
Inversion	Single	0.62	0.88	0.26
	Maximum	**0.77**	**0.94**	**0.41**
	Average	0.62	0.88	0.26

Table 5: Inverting GPT-4 paraphrases of human-text, an LLM *unseen* by the inversion model during training time. We generate 100 inversions per sample, and report the metrics achieved by a single inversion, by the best inversion (maximum), and the average across all inversions.

Model	BLEU
Phi-3	0.08
Mistral-7b	0.11
Llama-3-8B	0.08
GPT-4	0.23

Table 6: LLMs prompted to invert their own paraphrases both with, and without in-context examples. Generated 100 inversions per sample, best BLEU score per sample shown. Note that GPT-4 paraphrases already begin with a BLEU score of 0.21 (Table 5).

of human-written text is harder than paraphrases of machine-generated text, with the best scoring inversions achieving an average BLEU score of 25, which is half of that achieved when inverting paraphrases of machine-written text (§6.1). One reason paraphrases of human-written text may be harder to invert is that human-written words lie in the low-probability (i.e., low-rank) region of an LLM's predicted distribution, as observed by Gehrmann et al. (2019). (2) `output2prompt` does not recover significant portions of the original-text, we attribute this to its requirement of observing multiple outputs per prompt, and to the fact that the model has much lower capacity than ours (T5-base vs Mistral-7B).

6.3 Can inversion models invert a novel paraphraser?

To answer this question, we prompt GPT-4, an unseen LLM during training time, to paraphrase the human-written `Reddit` test set. We use our inversion model trained on Reddit to invert each paraphrase 100 times, and report the metrics in Table 5. Surprisingly, we find that **GPT-4 is easier to invert than the models seen during training**, with our model achieving a BLEU score of 41. We attribute this to GPT-4 paraphrases retaining more of the original text, with its paraphrases achieving a BLEU score of 21 in contrast to the BLEU score of 8 achieved by the LLMs used for training (Table 3).

6.4 Can an LLM invert its own paraphrases?

We prompt each LLM that generated a paraphrase in our `Reddit` dataset to invert its own paraphrase. We generate 100 inversions, and report the average maximum BLEU score achieved in Table 6. Overall, we find when prompted, state-of-the-art LLMs, including GPT-4, are unable to invert their own paraphrase. This implies that even if some parametric knowledge encodes the paraphrasing process, the LLM is not able to recover the original text given a paraphrase, further motivating our approach of training paraphrase inversion models.

7 Conclusion

Summary of findings In this paper, we presented the first detector-agnostic defense against paraphrase attacks. This defense relies on the novel task of *paraphrase inversion*, where the goal is to recover the original tokens of paraphrased text. Furthermore, we proposed two paraphrase detection schemes: one based upon a neural-classifier and another that relies on the number of edits our inversion model makes. When combined with one of the proposed paraphrase detectors, our pipeline improves the results of 7 machine-text detectors across 3 domains by an average of +22% `AUROC`. We attribute the effectiveness of our defense to the *stylistic* similarity of the inverted paraphrases to the original text, which is sufficient for machine text detectors to accurately classify the inverted text. Furthermore, we show that when our defense is trained on one domain, it generalizes to another, suggesting that paraphrasers exhibit consistent biases that can be exploited both for detecting paraphrased text and for learning to invert them.

Limitations

The number of paraphrases we use to train our inversion models is limited by our compute budget. We expect that training on additional LLM-generated paraphrases will improve all the results reported in the paper; as such, the results reported here should be viewed as a lower bound on achievable performance. Our compute budget also precluded experimenting with larger local models such as Llama-3 70B; however, we do include results with GPT-4 which is of comparable or greater quality.

Acknowledgements

Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DEAC52-07NA27344. This work was supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #D2022-2205150003. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

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 *ICLR*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. 2024. RAID: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12463–12492, Bangkok, Thailand. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 758–764.

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

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 *ICML*.

Julian Hazell. 2023. Spear phishing with large language models. *Preprint*, arXiv:2305.06972.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. *Advances in neural information processing systems*, 36:15077–15095.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 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*, pages 1808–1822, Online. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Aleem Khan, Elizabeth Fleming, Noah Schofield, Marcus Bishop, and Nicholas Andrews. 2021. A deep metric learning approach to account linking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5275–5287, Online. Association for Computational Linguistics.

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

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. *Advances in Neural Information Processing Systems*, 36:27469–27500.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Haoran Li, Mingshi Xu, and Yangqiu Song. 2023. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14022–14040, Toronto, Canada. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. 2024. Language model inversion. In *The Twelfth International Conference on Learning Representations*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Martin Potthast, Matthias Hagen, and Benno Stein. 2016. Author obfuscation: Attacking the state of the art in authorship verification. In *Conference and Labs of the Evaluation Forum*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Rafael A. Rivera-Soto, Olivia Elizabeth Miano, Juanita Ordonez, Barry Y. Chen, Aleem Khan, Marcus Bishop, and Nicholas Andrews. 2021. Learning universal authorship representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 913–919, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2025. Can ai-generated text be reliably detected? *Preprint*, arXiv:2303.11156.

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.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 377–390, New York, NY, USA. Association for Computing Machinery.

Rafael Alberto Rivera Soto, Kailin Koch, Aleem Khan, Barry Y. Chen, Marcus Bishop, and Nicholas Andrews. 2024. Few-shot detection of machine-generated text using style representations. In *The Twelfth International Conference on Learning Representations*.

Piotr Teterwak, Chiyuan Zhang, Dilip Krishnan, and Michael C. Mozer. 2021. Understanding invariance via feedforward inversion of discriminatively trained classifiers. *Preprint*, arXiv:2103.07470.

Andrew Wang, Cristina Aggazzotti, Rebecca Kotula, Rafael Rivera Soto, Marcus Bishop, and Nicholas Andrews. 2023. Can authorship representation learning capture stylistic features? *Transactions of the Association for Computational Linguistics*, 11:1416–1431.

Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA. Association for Computing Machinery.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Fan Wu, Emily Black, and Varun Chandrasekaran. 2025. Generative monoculture in large language models. In *The Thirteenth International Conference on Learning Representations*.

Collin Zhang, John Xavier Morris, and Vitaly Shmatikov. 2024a. Extracting prompts by inverting LLM outputs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14753–14777, Miami, Florida, USA. Association for Computational Linguistics.

Yiming Zhang, Avi Schwarzschild, Nicholas Carlini, J Zico Kolter, and Daphne Ippolito. 2024b. Forcing diffuse distributions out of language models. In *First Conference on Language Modeling*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

A Subsampling the Reddit Dataset

We subsample the dataset to authors who post in `r/politics` and `r/PoliticalDiscussion`, keeping comments composed of at least 64 tokens but no more than 128 tokens according to the LUAR tokenizer. Furthermore, we remove authors with less than 10 comments, and randomly sample 10 comments from all others, ensuring that no author is over-represented.

To learn to invert paraphrases, we must observe a diverse set of source documents and corresponding paraphrases. However, a random sample of documents may not provide broad enough coverage of writing styles. For example, when we prompt GPT-4 to generate a paraphrase of "HELLO WORLD", it produces "Greetings, Universe!", removing the capital letters. Without observing authors who write only with capital letters during training, it would be impossible for the inversion model to invert the paraphrase. As such, we split authors into training, validation, and testing splits by sampling authors evenly across the *stylistic* space. We use `LUAR` (Rivera-Soto et al., 2021), an embedding that captures stylistic features, to embed each author's posts into a single stylistic embedding. Then, we cluster the dataset using K-Means, setting $K = 100$. Finally, we take 80% of the authors from each cluster for training, 10% for validation, and randomly sample 100 authors (2,449 posts) of those remaining for testing.

B Creating Datasets from RAID

In contrast to our `Reddit` dataset, the RAID (Dugan et al., 2024) benchmark doesn't contain author-labels. Therefore, sampling authors evenly across stylistic space as in §4.1 is not possible. RAID contains paraphrases of machine-text using DIPPER, but lacks paraphrases of human-text. To address this, we paraphrase all human-text within `ArXiv` and `MovieReviews` with DIPPER, using the same hyper-parameters as the creators of RAID (60 lexical diversity, 0 order diversity, 512 max-tokens). We pair up the machine-text with their corresponding paraphrases, randomly sampling 80% of these pairs for training, 10% for validation, and 10% for testing. Furthermore, ensure that the validation sets contain an equal number of machine-text and paraphrases of machine-text, augmenting them with an equal number of the human-paraphrases we generated. We follow the same procedure for test set, while additionally mensuring that we have exactly 500 samples for each category: human-text, paraphrases of human-text, and paraphrases of machine-text. The validation sets are used to train the edit-based detector discussed in §3.3, while the training sets are used to train both our paraphrase inversion and paraphrase detection models.

C Ablations

How does varying the sampling procedure impact paraphrase inversion? In Table 7 we show the effect that the decoding temperature has in the quality of the inversions generated by our untargeted inversion model. We generate 100 inversions for every paraphrase in our test dataset, and report metrics using the “max" scoring strategy dicussed in §3. We observe that temperature plays an important role in the quality of the inversions, with values too low or too high significantly degrading the quality of the inversions. As the temperature increases, the entropy of the distribution approximates that of a uniform distribution, thereby diffusing the style of the inversions. Conversely, as the temperature decreases, the inversion model becomes over-confident in its predictive distribution, thereby not exploring neighboring tokens and styles.

Temperature	Style Sim.	BLEU
0.3	0.67	0.23
0.5	0.69	0.24
0.6	0.70	0.25
0.7	0.70	0.25
0.8	0.71	0.24
0.9	0.71	0.23
1.5	0.55	0.06

Table 7: Effect of the temperature in the quality of the untargeted inversions.

Training Temperature	Style Sim.	BLEU
0.3	0.71	0.26
0.5	0.70	0.25
0.7	0.70	0.25

Table 8: Effect of training on a paraphrase dataset generated with different temperature values.

Are paraphrases generated with lower temperature values easier to invert? To answer this question, we re-generate our human-text paraphrase data with lower temperature values, training and testing the untargeted inversion model in matched temperature conditions. We report the results in Table 8. We observe that, as the temperature decreases, the similarity metrics improve. We attribute this to the LLMs becoming over-confident in their predictive-distribution, thereby generating less diverse data which in turn is easier to invert.

D Prompts

D.1 Paraphrasing

When paraphrasing with an instruction-tuned LLM, we use the following prompt:

Prompt:

```
Rephrase the following passage:
<PASSAGE>
Only output the rephrased-passage,
do not include any other details.
Rephrased passage:
```

We also clean out all obvious generation artifacts, keeping only the paraphrased text.

D.2 Inversion

D.2.1 Inversion

Prompt:

```
[INST] The following passage
is a mix of human and machine
text, recover the original
human text:  {generation}
[/INST]\n#####\nOutput:
{original}
```

D.3 Prompting Inversion

Prompt:

```
The following passage is a mix of
human and machine text, recover
the original human text:
```

D.4 Generating Reddit Responses

Prompt:

```
Write a response to the following
Reddit comment:  comment
```

E Dataset Statistics

We show the statistics of the `Reddit`, `RAID-ArXiv`, and `RAID-MovieReviews` in Table 9.

F Training Hyper-Parameters

We train all our inversion models with the hyper-parameters shown in Table 10. We train all our models on 4 NVIDIA-A100 GPUs. Each model took at most 10 hours to train.

Most of the compute was spent generating the inversions necessary to run all the experiments, which are in the ballpark of 1M total generations. We used VLLM (Kwon et al., 2023) to speed up the inference time. We estimate an upper bound of around 150 GPU hours to run all experiments.

Split	Number of Examples
Reddit Human-Paraphrase	
Train	204260
Valid	24549
Test	2449
Reddit Machine-Paraphrase	
Train	239710
Valid	28883
Test	2854
Reddit Machine-Text Detection	
Test	1500
RAID-ArXiv	
Train	48035
Valid	3798
Test	1500
RAID-MovieReviews	
Train	25649
Valid	1329
Test	1500

Table 9: Statistics of the `Reddit`, `RAID-ArXiv`, and `RAID-MovieReviews` datasets.

Hyper-Parameter	Value.
Learning Rate	$2e^{-5}$
Number of Epochs	4
LoRA-R	32
LoRA-α	64
LoRA-Dropout	0.1

Table 10: Training Hyper-parameters.

G Cleaning Obvious Machine-Generated Artifacts

When creating our dataset with the paraphrase prompt in Appendix D, we noticed that the LLMs prompted oftentimes introduced obvious generation artifacts that could be used to easily invert the paraphrases. To ensure that our inversion models don't rely on any shortcuts, we manually inspected the dataset to find the obvious artifacts and wrote the following code to remove them from the text: `https://github.com/rrivera1849/inversion/blob/main/src/utils_clean_gen.py`