Appendix

A Ethical Considerations

Our goal in this paper is not to provide a recipe for potential attackers (e.g., college students wishing to use ChatGPT in their essays) to evade AI text detection systems. Rather, we wish to bring awareness to the wider community about the vulnerabilities of current AI-generated text detectors to simple paraphrase attacks. These detectors are not useful in their current state given how easy they are to evade. We encourage the research community to stress test their detectors against paraphrases, and to develop new detectors which are robust against these attacks. To facilitate such research, we open source our paraphraser and associated data / code.

Furthermore, we propose not just an attack but also a potentially strong defense against this attack. Our detection strategy is simple, relying on retrieval over a corpus of previously-generated sequences. We empirically show that such a detection algorithm could work at scale and provide extensive discussion on possible methods to improve performance (Appendix B.2), as well as discussing possible limitations and approaches to tackling them (Appendix B.1). We hope that retrieval-based AI-generated text detectors rapidly improve and are eventually deployed in conjunction with other detection methods like watermarking / classifiers.

B Limitations of retrieval-based detection and ideas for scaling it further

In Appendix B.1, we first point out some limitations of using retrieval for AI-generated text detection (Section 5), some of which potentially apply to all existing detectors. Along with limitations, we provide several possible workarounds. In Appendix B.2, we then discuss ideas that can make the proposed retrieval detection work well at an even larger scale than the one we discussed in Section 5.

B.1 Limitations of retrieval for detection

While retrieval over previously-generated sequences is an effective defense against paraphrase attacks, it also suffers from key limitations, some of which apply broadly to all existing detectors. We discuss these limitations below and discuss possible solutions:

1. **Detection is specific to an API**. Unlike other general-purpose AI detection algorithms e.g. OpenAI's classifier [OpenAI, 2023], retrieval can only detect generations from the API over which the database is built. API #1 has no access to the database of generations from API #2, and thus will not be able to detect generations produced by API #2.
2. **The API provider needs to provide a retrieval infrastructure**. After the release of ChatGPT [Schulman et al., 2022], AI chatbots are getting widespread adoption. At a conservative rate of 5M queries a day, the database will have almost two billion entries in a year. Complex retrieval infrastructure (like modern search engines) will be necessary to retrieve over these large databases with low latency.
3. **False positives due to training data memorization**. Language models have been shown to memorize sequences verbatim from their training data [Carlini et al., 2021], such as the Gettysburg Address [Radford et al., 2019]. Despite being originally written by humans, these sequences will be classified as model-generated by our detector. To tackle this issue, we suggest API providers additionally perform retrieval over the training data used to train the model. If a sequence is found in the training set as well as the generation database, it is likely to be an instance of training set memorization.
4. **Privacy concerns.** Providing a retrieval detection service partially exposes the database of previously generated text by *all* users. This raises concerns of membership inference attacks [Shokri et al., 2017] on private user data which may appear in the generated text. To mitigate this, we suggest: (1) users should be encouraged not to provide any sensitive private data in their prompts to APIs, a practice already followed by ChatGPT[10] and Bard[11]; (2) API providers only provide a binary output from this detector (AI-generated or not), rather than actual search results; and (3) API providers rate-limit queries from IP addresses.

[10] https://chat.openai.com

[11] https://bard.google.com

5. **Slight reduction in accuracy with large databases.** As we observed in Section 5.3, the accuracy of detecting paraphrased text slightly degrades as the database of retrievals gets larger. However, we found this decrease to be quite small (only 1% on PG19 scaling 1M generations to 15M), despite using fairly primitive retrievers like BM25. Moreover, unperturbed AI-generated text will always be detected with 100% accuracy using our method, irrespective of corpus size.
6. **Tasks with constrained output space or short outputs**. Similar to all other detection algorithms, it may be hard or even impossible to distinguish AI-generated outputs for tasks with a constrained output space (like sentence-level translation, classification) or very short outputs (as shown in Section 5.3). Thus, we believe the main utility of AI-generated text detection is for longer-form generated text, and hence we focus on tasks like long-form QA and open-ended text generation with relatively lengthy outputs. Note that to avoid detection, a sophisticated attacker may try to generate long-form text in smaller chunks using multiple API calls, where each newly-generated chunk is incrementally concatenated to the prompt. This is not a concern for our method if retrieval is done over the corpus of prompts concatenated with generations.
7. **Iterative attacks with access to detector.** A final concern is that attackers with access to detection algorithms will iteratively modify their perturbations until they avoid detection. While this is a valid concern for all detectors, we believe retrieval has an important advantage over the alternatives. Since the corpus of previously-generated text is proprietary, only the API provider can provide access to this detection service - it is impossible for attackers to locally reproduce this detector. This allows API providers to adopt several mitigation strategies such as (1) rate-limiting queries to avoid iterative attacks; (2) providing retrieval access only to verified users (e.g., teachers); and (3) detecting possible iterative attacks by analyzing previously queries to the retriever.

B.2 Ideas to make retrieval detection work well at an even larger scale

In Section 5.3, we observed that our proposed retrieval detector is effective even with a large corpus of 15M previously-generated sequences. While we do not have access to a larger corpus of generations (billion-scale), in this section we describe some ideas to improve retrieval detection at such a scale.

1. **Timestamp filtering in retrieval corpus.** To reduce the large search space, the detector interface could provide users with an option to restrict retrieval to only a fixed time period during which the text was likely to be generated. For instance, a common use-case of AI-generated text detection might be when teachers attempt to catch plagiarism in college essays. Teachers could restrict retrieval to only those generations created during the assignment window.
2. **More sophisticated retrieval strategies.** In our work, we only explore simple retrieval strategies like BM25. However, several more sophisticated retrieval strategies exist, which are known to boost performance [Thakur et al., 2021] and could be useful here. These include methods like re-ranking of top-k retrievals [Khattab and Zaharia, 2020] or dense retrieval [Karpukhin et al., 2020]. We do note that these more complex methods are also slower, and latency is likely to be a pressing concern for API providers.
3. **Fine-tuning dense retrievers for the detection task.** The retrievers in our work are not fine-tuned for the task of AI-generated text detection. However, we hypothesize that fine-tuning retrievers on this task can help retrievers adapt better to the retrieval corpus and detection task. Specifically, a contrastive learning approach could be adopted here: positive pairs are paraphrased or otherwise noised sequences paired with their generations, while negative pairs are human-written continuations paired with the machine-generated text.

C Experiments measuring intrinsic paraphrase generation quality

Our experiments in Section 4 and Section 5 focused on attacking AI-generated text detectors with paraphrases and defending against these paraphrase attacks. We used DIPPER as the underlying paraphrase generation model for all of these experiments. Are DIPPER's paraphrases actually good enough to make the attack worthwhile, and can simpler paraphrasers be just as effective as DIPPER? In this section, we conduct careful ablation experiments (Appendix C.1) and human evaluations (Appendix C.2) to validate the effectiveness of DIPPER at preserving the semantics of the input generation. Our results show that DIPPER effectively leverages surrounding context to paraphrase multiple sentences while preserving input semantics.

Table 3: Ablation experiments demonstrate the high quality of DIPPER's paraphrases compared to alternatives. Displayed scores are the percentage of cases in which rewrite A is preferred over B by one of the three metrics, with subscripts showing absolute average scores on each metric across the dataset. Overall, DIPPER benefits from context outside the input (Experiment 1), multi-sentence paraphrasing (Experiment 2), and is not too far behind non-paraphrased text in terms of quality (Experiment 3).

Open-ended generation with GPT2-XL on Wikipedia prompts						
	RankGen-XL		GPT3.5 davinci-003 perplexity		unigram overlap with prompt	
Control	rewrite A	rewrite B	rewrite A	rewrite B	rewrite A	rewrite B
Experiment 1: *Is context helpful for paraphrasing?* rewrite A = DIPPER with context rewrite B = DIPPER no context						
20L	**65%** $_{10.2}$	35% $_{9.2}$	**71%** $_{11.5}$	29% $_{12.6}$	**55%** $_{41.3}$	45% $_{40.7}$
40L	**64%** $_{9.8}$	36% $_{8.5}$	**70%** $_{11.9}$	30% $_{13.0}$	**57%** $_{40.7}$	43% $_{39.9}$
60L	**67%** $_{9.6}$	33% $_{7.6}$	**68%** $_{12.3}$	32% $_{13.6}$	**56%** $_{39.9}$	44% $_{39.2}$
60L,60O	**65%** $_{8.3}$	35% $_{6.4}$	**75%** $_{12.9}$	25% $_{15.0}$	**58%** $_{39.4}$	42% $_{38.2}$
Experiment 2: *Is it helpful to paraphrase multiple sentences at a time?* rewrite A = DIPPER 3 sentences at a time rewrite B = DIPPER 1 sentence at a time						
20L	**58%** $_{9.2}$	42% $_{8.6}$	**86%** $_{12.6}$	14% $_{15.3}$	48% $_{40.7}$	**52%** $_{40.9}$
40L	**56%** $_{8.5}$	44% $_{8.1}$	**83%** $_{13.0}$	17% $_{15.8}$	45% $_{39.9}$	**55%** $_{40.4}$
60L	**54%** $_{7.6}$	46% $_{7.5}$	**79%** $_{13.6}$	21% $_{15.7}$	45% $_{39.2}$	**55%** $_{39.9}$
60L,60O	**50%** $_{6.4}$	**50%** $_{6.4}$	**85%** $_{15.0}$	15% $_{19.6}$	42% $_{38.2}$	**58%** $_{39.5}$
Experiment 3: *Does paraphrasing preserve the quality of the original text?* rewrite A = no paraphrasing rewrite B = DIPPER						
20L	**50%** $_{10.4}$	**50%** $_{10.2}$	**61%** $_{11.1}$	39% $_{11.5}$	**51%** $_{41.6}$	49% $_{41.3}$
40L	**57%** $_{10.4}$	43% $_{9.8}$	**67%** $_{11.1}$	33% $_{11.9}$	**55%** $_{41.6}$	45% $_{40.7}$
60L	**58%** $_{10.4}$	42% $_{9.6}$	**73%** $_{11.1}$	27% $_{12.3}$	**58%** $_{41.6}$	42% $_{39.9}$
60L,60O	**68%** $_{10.4}$	32% $_{8.3}$	**79%** $_{11.1}$	21% $_{12.9}$	**61%** $_{41.6}$	39% $_{39.4}$

C.1 Ablation studies on DIPPER

In this section, we perform automatic evaluations to confirm the efficacy of DIPPER as a paraphraser. From a survey of existing paraphrasers that we carry out in Appendix D.1, DIPPER possess two unique features that differentiate it from other paraphrasers: (1) its ability to leverage context from *outside* of the text to be paraphrased (such as the prompt); and (2) its ability to paraphrase multiple sentences at once. How useful are these features while paraphrasing long sequences of text?

To answer this question, we first train an ablated version of DIPPER by constructing a training dataset (Section 3) without any left or right context, and then fine-tuning T5-XXL using the same hyperparameters as in Section 3. We call this model DIPPER-no-ctx. We paraphrase 1K open-ended generations from GPT2-XL using both DIPPER and DIPPER-no-ctx, using each of the four configurations of diversity control codes studied in this paper. We then evaluate the quality of the paraphrased text using three metrics: (1) GPT3.5-davinci-003 perplexity [Brown et al., 2020] of the prompt concatenated with the paraphrased continuation; (2) RankGen compatibility between the prompt and the paraphrased continuation [Krishna et al., 2022a]; and (3) unigram token overlap between the paraphrased continuation and the prompt.

Contextual paraphrasing leads to higher quality paraphrases. In Table 3 (Experiment 1), we observe that across all four control code configurations and all three metrics, paraphrases from DIPPER are preferred over paraphrases from DIPPER-no-ctx. Specifically, with the lexical and order control codes set to 60% (most diverse), DIPPER paraphrases are preferred by GPT3.5 perplexity 75% of the time compared to non-contextual paraphrases (average perplexity drop of 12.9 vs 15.0).

Table 4: This table shows how often each point in the Likert scale was chosen by 3 annotators for the pairs of original and paraphrased texts. Twenty text pairs are randomly selected for each lexical code (L). 81.8% of the time, our model DIPPER provides a paraphrase which is nearly equivalent to the input in terms of semantic meaning.

L	Sum of 4 and 5	5 Approx. equivalent	4 Nearly equivalent	3 Somewhat equivalent	2 Topically related
20	95.0%	63.3%	31.7%	5.0%	0.0%
40	78.3%	45.0%	33.3%	21.7%	0.0%
60	70.0%	28.3%	41.7%	28.3%	1.7%
Total	81.1%	45.6%	35.6%	18.3%	0.6%

Paraphrasing multiple sentences at a time is better than paraphrasing individual sentences. Next, we use our DIPPER-no-ctx model to compare two settings: paraphrasing 3 sentences at a time vs paraphrasing 1 sentence at a time before concatenating. We hypothesize that the former will produce higher quality paraphrases since we expect it to better connect discourse elements across the text. Indeed, in Table 3 (Experiment 2) across all control codes, GPT3.5 and RANKGEN usually prefer multi-sentence paraphrases over the single-sentence baseline. This preference is 79% or higher for all control codes when evaluating with GPT-3.5 perplexity, reaching 85% for L60,O60.

DIPPER paraphrases are close to the unperturbed GPT-2 XL generations. Finally, we compare DIPPER with the original GPT2-XL generations (without paraphrasing) on the same three metrics. While we expect metrics to prefer non-paraphrased text, a strong paraphraser will produce text that is close to the original in terms of these metrics. Table 3 (Experiment 3) confirms our hypothesis: at L20, RANKGEN has a 50-50 preference between the two outputs, while GPT3.5 prefers the non-paraphrased generations just 61% of the time, with an average perplexity gain of just 0.4 (11.1 to 11.5). At more diverse control codes, preference for GPT2-XL generations does go up (58% RANKGEN, 73% GPT3.5 for L60), but absolute scores continue to be close (11.1 vs 12.3 GPT-3.5 perplexity). Note that while all of these ablations use just a single paraphrase sample, it is easy for an attacker to obtain multiple samples from DIPPER and choose the sample that maximizes these metrics (as discussed in Section 4.3).

C.2 Human evaluation of semantic preservation using DIPPER

The automatic semantic similarity scores in Table 1 and 3 indicate that DIPPER generates paraphrases that are faithful to the original input paragraphs. To confirm this result with human evaluation, we hire three native English teachers and/or editors on Upwork[12] to evaluate the semantic fidelity of the paraphrases. As human evaluation is expensive, we fix the order diversity (O) to be 0 and focus on the impact of the lexical diversity. We evaluate paraphrases with the lexical codes $L20$, $L40$, and $L60$, corresponding to moderate, medium, and high lexical diversity. Twenty paraphrases are sampled randomly for each lexical code, resulting in 60 original text and paraphrase pairs.

The evaluation is conducted on the platform Label Studio [Tkachenko et al., 2020-2022].[13] As shown in the interface of our annotation platform Figure 7, the text to be paraphrased (highlighted in yellow) are preceded by its context. The annotators see the same amount of text as DIPPER. They need to first read the texts, select one point on the Likert scale, then provide free-form comments justifying their ratings. We estimated that the evaluation of each paraphrase takes 1.5 to 2 minutes. As such, we pay $15 as a base rate with a bonus for the reasonable extra time that the annotators spend on the tasks.

Among the 60 original text and paraphrase pairs, the three annotators agreed on their choice 28.3% of the time, and 60% of the time the point they chose on the scale differs by 1. Table 4 reports how often each point on the Likert scale is chosen. Over 80% of the time, our annotators rate DIPPER's paraphrases as nearly equivalent (4 out of 5) or approximately equivalent (5 out of 5).

A qualitative analysis of the free-form annotator comments reveals systemic strengths and shortcomings of DIPPER. Table 5 provides two representative examples for each lexical code that is evaluated in our human study.

[12] https://www.upwork.com
[13] https://labelstud.io/

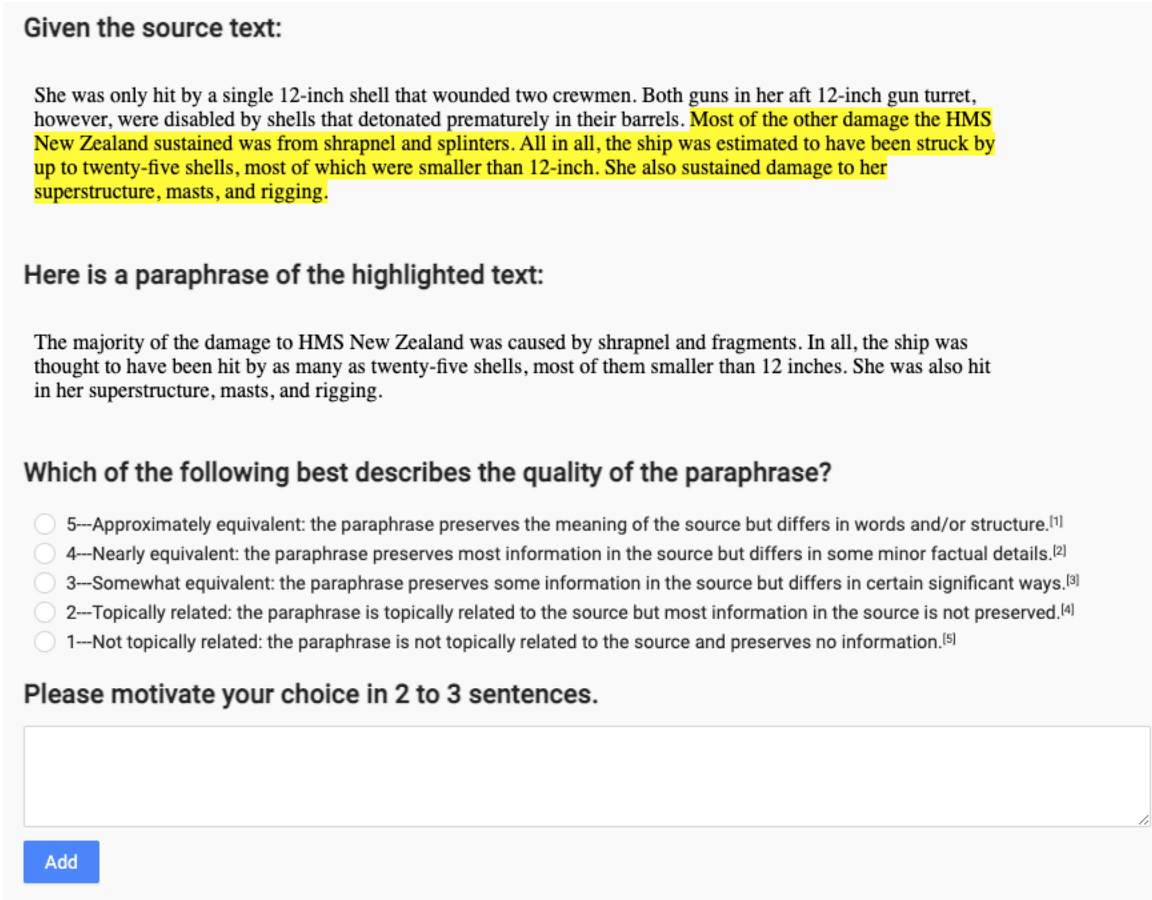

Figure 7: The interface of the annotation platform used in our human study

Strengths First, the third example in Table 5 exemplifies DIPPER's ability to leverage information from context to increase diversity while maintaining coherence (i.e., from *line... reference the song's title* to *reference to "I'm the Greatest"*). The same is observed in row 2 where DIPPER uses the context to interchange *he* and *Churchill*. A paraphrase model without looking into context will have great difficulty in doing this and no prior paraphraser (see Table 6 for a list) is capable of that. Second, the example in the fifth row highlights DIPPER's ability to make significant changes to original texts with a high lexical diversity code ($L60$) (see the color coding) while preserving their semantic meaning as rated by the annotators.

Qualitative shortcomings: The first shortcoming is that, when the original text contains new created proper names (unlike common people and country names), such as the ones in row 6 (*Homing Attack* and *Slide Attack*), a high lexical code has a tendency to change such nouns, leading to the result that one of our annotators deems it to be only topically related to the original. However, this shortcoming can be overcome by decreasing the lexical code, which a user can choose from a continuous range (from 0 to 100). For instance, in row 1 with `lex=20`, the songs' names *M's Confession* and *Gone Fishing* are kept intact. Another shortcoming is that DIPPER occasionally omits content from an original text. While in some cases such removal is acceptable (see row 6), in other cases it causes significant change in the meaning of the text (see row 4). However, the former case can be overcome by paraphrasing a shorter paragraph at a time.

Overall, the human study shows that DIPPER performs well at preserving the semantic meaning of original texts while introducing both semantic and syntactic diversity. Because DIPPER provides user-friendly controllabilty of output diversity, a user can adjust the control code to find the most suitable paraphrase for their need.

D Related work for discourse paraphrasing

D.1 Survey of paraphrase generation papers

As an important NLP task, paraphrasing has attracted much attention. Many models have been proposed to improve the quality of paraphrases. To position our model DIPPER and highlight its strengths, we conduct a survey of paraphrase generation papers from 2018 to 2022 (Table 6) and focus on the following four aspects:

1. Whether a model can paraphrase a paragraph at once,
2. whether a model can merge or split consecutive sentences when appropriate,
3. whether a model leverages context surrounding an input sentence when paraphrasing,
4. whether a model provides control knobs for users to customize the output diversity.

The survey shows that only three out of 25 papers mentioned that their model can paraphrase more than one sentence (but not necessarily at once). None of them enables their model to merge or split sentences when paraphrasing. No model uses information from context surrounding an input sentence during inference time. Finally, 14 papers offer ways for users to customize the diversity of paraphrases. However, most diversity control methods such as constituency parses or exemplars may not be straightforward and intuitive to end-users as the scalar control knobs in DIPPER.

In contrast to the papers in the survey, DIPPER nicely combines all desiderata into one model and offers intuitive control knobs for lexical and syntactic diversity. Automatic and human evaluation show that DIPPER can efficiently leverage context information and reorganize sentences while having high fidelity in meaning (Appendix C).

D.2 Other related work

In this section we discuss a few additional less related papers which were not included in our survey in Appendix D.1. Our discourse paraphraser is closely related to work on contextual machine translation, where source/target context is used to improve sentence-level machine translation [House, 2006, Jean et al., 2017, Wang et al., 2017, Tiedemann and Scherrer, 2017, Kuang et al., 2018, Agrawal et al., 2018, Miculicich et al., 2018, Zhang et al., 2018, Xiong et al., 2019, Jean et al., 2019, Voita et al., 2019a, Yin et al., 2021, Mansimov et al., 2021]. Prior work has shown that context helps with anaphora resolution [Voita et al., 2018], deixis, ellipsis, and lexical cohesion [Voita et al., 2019b]. Efforts to make paraphrase generation more contextual have been quite limited. A few efforts have attempted to use sentence level context to paraphrase phrases [Connor and Roth, 2007, Max, 2009], and dialogue context to paraphrase individual dialogues in a chat [Garg et al., 2021].

Our work is also related to efforts in text simplification to go beyond a sentence, by collecting relevant datasets [Xu et al., 2015, Devaraj et al., 2021] and building unsupervised algorithms [Laban et al., 2021]. Note that our work focuses on a general-purpose paraphrasing algorithm and is not tied to any particular style, but could be utilized for document-level style transfer using techniques like Krishna et al. [2020, 2022b]. Similar efforts have also been undertaken in machine translation, [Popescu-Belis et al., 2019, Junczys-Dowmunt, 2019, Maruf et al., 2021], attempting to translate paragraphs/documents at once.

E More background on detectors of AI-generated text

In this section, we provide an overview of existing algorithms that have been developed for the purpose of detecting machine-generated text. Such algorithms fall into three main categories: (1) watermarking algorithms, which modify the generative algorithm to encode hidden information unique to the API (Appendix E.1); (2) statistical outlier detection methods, which do not modify the generative algorithm but look for inherent artifacts in generated text (Appendix E.2); and (3) classifiers trained to discriminate machine-generated text from human-written text (Appendix E.3). Finally, in Appendix E.4, we compare and contrast our work to Sadasivan et al. [2023], who also note the efficacy of paraphrasing attacks but do not consider a retrieval-based defense in their pessimistic conclusion about the fate of AI-generated text detection.

E.1 Watermarking language model outputs

A "watermark" is a modification to the generated text that can be detected by a statistical algorithm while remaining imperceptible to human readers. Effective watermarks are difficult to remove and have little effect on the quality of generated text. Prior work attempted to watermark natural language using syntax tree manipulations [Topkara et al., 2005, Meral et al., 2009], and this area has gotten renewed interest with large language models generating human-like text [Abdelnabi and Fritz, 2021, Grinbaum and Adomaitis, 2022]. Most recently, Kirchenbauer et al. [2023] propose a simple algorithm that only requires access to the LLM's logits at each time step to add watermarks. The watermark can then be verified with only blackbox access to the LM and knowledge of a specific hash function. This algorithm operates in three steps:

1. **Mark a random subset of the vocabulary** as "green tokens" (or tokens representing the watermark, as shown in Figure 1) using the hash of the previously generated token as a random seed. A total of $\gamma|V|$ tokens are marked green where γ is the fraction of the tokens that are watermarked with default $\gamma = 0.5$.
2. **Increase the logit value** for every green token by a constant δ ($= 2$ by default), which denotes the watermark strength. This raises the probability of sampling green watermarked tokens, especially for high-entropy distributions.
3. **Sample sequences** using decoding algorithms such as nucleus sampling [Holtzman et al., 2020], leveraging the modified probability distribution at each timestep before truncation.

Detecting the watermark: Verifying whether a text is generated by a watermarked LM is possible with just knowledge of the hash function and tokenizer. Specifically, the verifier tokenizes the text and counts the number of green tokens it contains. This is used to calculate the standard normal score (z-score) for the hypothesis test. If the sequence with T tokens contains a certain number of the green token (denoted as $|s|_G$), the z-score can be computed by:

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

Intuitively, a higher z-score implies it is less likely for a human to have written the text (null hypothesis) since it contains a higher than expected number of green tokens. Kirchenbauer et al. [2023] recommend using a high z value ($z > 4$, or $p < 3 \times 10^{-5}$) to reduce the risk of false positives (human-written text classified as AI-generated). Low false positive rates are critical in AI-generated text detection algorithms [OpenAI, 2023]—we discuss this in Section 4.1.

E.2 Statistical outlier detection methods

Unlike the watermarking algorithms, outlier detection algorithms make no modification to the generative algorithm. Instead, they attempt to distinguish between human-written and machine-generated text based on the presence of artifacts in generated text [See et al., 2019, Holtzman et al., 2020]. Early methods detect statistical irregularities in measures such as entropy [Lavergne et al., 2008], perplexity [Beresneva, 2016], and n-gram frequencies [Grechnikov et al., 2009, Badaskar et al., 2008]. After the release of GPT-2, Gehrmann et al. [2019] introduced the GLTR visualization tool to assist human verifiers in detecting machine-generated text. Most recently, the release of ChatGPT has prompted the development of two new tools, namely a closed-source tool called GPTZero [Tian, 2023], and open-source DetectGPT [Mitchell et al., 2023]. DetectGPT uses an observation that model-generated text lies in the negative curvature regions of the model's log probability function. It constructs multiple perturbations of the model generated text (using a mask-and-fill strategy), and compares the log probability of the perturbations with the unperturbed generation. Text is considered model generated if the log probability of the unperturbed text is significantly higher than the log probability of perturbations.

E.3 Classifiers

The third class of detection methods relies on classifiers that are fine-tuned to distinguish human-written text from machine-generated text. Early efforts in this vein use classifiers to detect fake reviews [Hovy, 2016] and fake news [Zellers et al., 2019]. Other related studies examine classification performance across domains [Bakhtin et al., 2019] and decoding strategies [Ippolito et al., 2020].

Such studies inspired others to use their insights to improve generative performance [Deng et al., 2020, Krishna et al., 2022a]. Most recently, OpenAI fine-tuned a GPT model to perform this discrimination task and released it as a web interface [OpenAI, 2023]. They fine-tuned this classifier using generations from 34 language models, with text sourced from Wikipedia, WebText [Radford et al., 2019], and their internal human demonstration data.

E.4 Comparison to Sadasivan et al. (2023)

In very recent concurrent work, Sadasivan et al. [2023] also demonstrate the utility of paraphrasing attacks against AI-generated text detectors. While their work makes use of off-the-shelf sentence-level paraphrase models, DIPPER possesses advanced discourse-level rewriting capabilities as well as fine-grained diversity control, which allows us to thoroughly analyze the effectiveness of various paraphrasing strategies. Our experiments also encompass more tasks, datasets, and detection algorithms. Moreover, we evaluate larger language models like GPT3.5-davinci-003. Finally and most importantly, our retrieval-based defense *directly contradicts* the "impossibility result" of Sadasivan et al. [2023] and its associated proof, which states that even an optimal detector will approach the performance of a random classifier as the distance between the distributions of LLM-generated text and human generated text goes to zero. Since our detector does not rely on properties of the text but rather a corpus search, the quality of the generated text is irrelevant to the effectiveness of our detector, and thus their proof does not apply to our method.

F More experimental details of our attack experiments

F.1 Details for training our paraphraser DIPPER

Our paraphraser DIPPER is a sequence-to-sequence Transformer neural network [Vaswani et al., 2017], initialized with the T5-XXL 1.1 checkpoint [Raffel et al., 2020] and fine-tuned on our paraphrase generation data, using early stopping on validation loss for held-out novels. We find it helpful to paraphrase a maximum of 3 consecutive sentences at time, which leads to better adherence to control codes. Our models are implemented in JAX [Bradbury et al., 2018] using the T5X library [Roberts et al., 2022] with the default fine-tuning hyperparameters. Training was done on 32 cloud TPUv3 chips, and took 6-12 hours to complete. At inference time, we use nucleus sampling [Holtzman et al., 2020] with $p = 0.75$ and a variety of control codes.

To make our paper more intuitive, we have slightly modified the notation that our actual pretrained model uses. Our pretrained model uses control codes $100 - L$ and $100 - O$, denoting lexical/order *similarity* rather than diversity. Also, `<sent>` is used instead of `<p>`. We will clearly document this in the code release.

F.2 Long-form question answering data processing

In Section 4 evaluate long-form question answering [Fan et al., 2019], in which an LM must answer a how/why question (e.g., *Why are almost all boats painted white?*) with a 250-350 word answer. To build a long-form question answering dataset, we scrape questions from the r/explainlikeimfive subreddit posted between July to December 2021.[14] We randomly sample 500 questions from each of six popular domains on the subreddit (biology, physics, chemistry, economics, law, and technology) and pair each question with its longest human-written answer, which yields 3K long-form QA pairs.

G Controlled comparisons of retrieval with other AI-generated text detectors on open-ended text generation

We conduct a controlled comparisons of retrieval on the open-ended text generation task with Wikipedia prompts (see Section 5.2). The result of the experiment is presented in Table 7.

[14] We choose this period since current language models have been trained on internet data available before June 2021 [OpenAI, 2022], this prevents verbatim copying from training data.

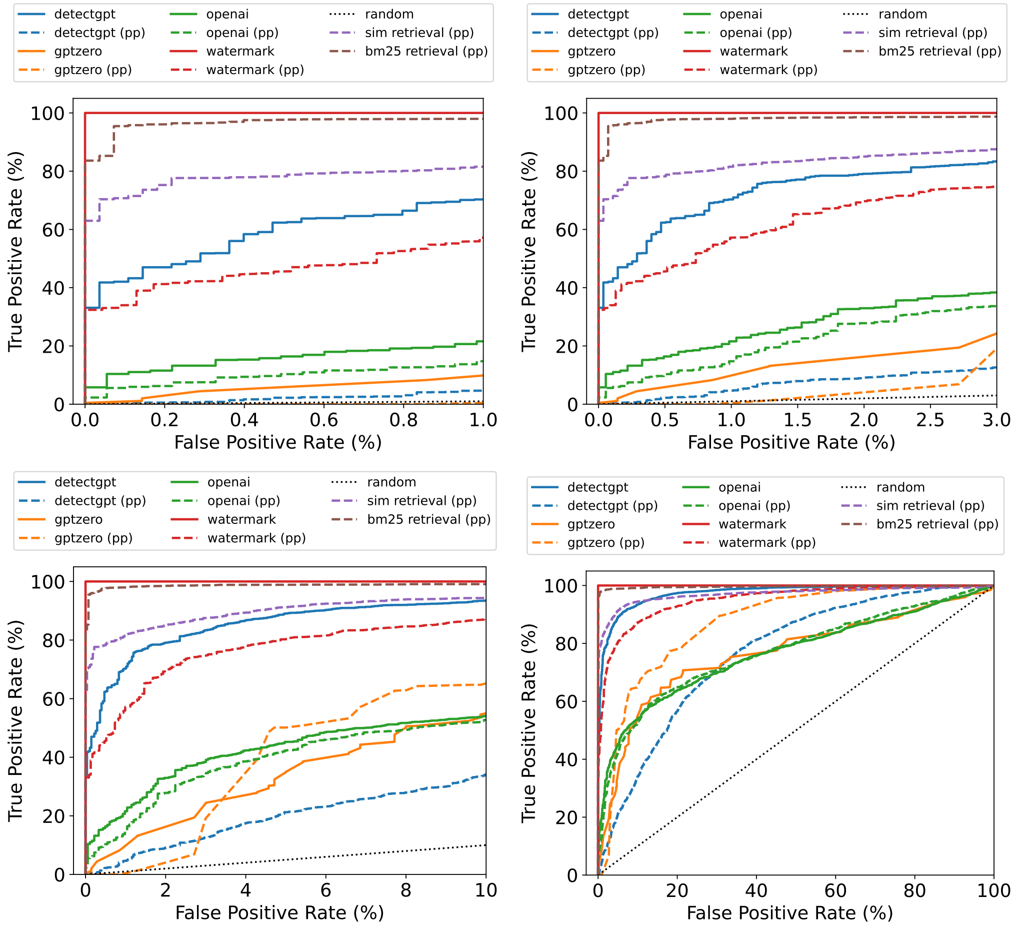

Figure 8: ROC curves for text generated by GPT2-XL, before paraphrasing (solid lines) and after paraphrasing (dashed lines, pp). Different plots represent different clipping thresholds on the X-axis.

971 # H ROC curves at different FPR

972 See Figure 8.

Table 5: Representative model outputs of each lexical code with Likert ratings and comments from the annotators. The texts in bold in the original texts are the context. Red words are the content being changed in the original text and green words are the changed content in the paraphrases.

L	Original	Paraphrase	Annotator Comment
20	**The first, originally used as Renard's theme, is heard during the mammoth "Antonov" cue on the recording, and is written for piano. The second new theme, used in the "Christmas in Turkey" track of The World Is Not Enough, is reused in the "Going Down Together" track.** Both themes were later adapted by David Arnold and featured in the film's score. The first theme appears in the cue "M's Confession" and the second theme appears in the cue "Gone Fishing".	Both themes were subsequently adapted by David Arnold and used in the film score. The first theme is used in the "M's Confession" cue, and the second in the "Gone Fishing" cue.	**5—Approx. equivalent** While the paraphrase is worded in a more simplistic way than the original and has shortened clauses, the information presented in both is the same.
20	**Churchill argues that the United States instituted blood quantum laws based upon rules of descendancy [. . .]** He has argued that the United States has used them to strip Indigenous people of their rights, land and culture in order to benefit those in power. He contends that these laws were designed to create a divide among Indigenous people, making them easier to control and manipulate. Churchill has further argued that this manipulation continues today, as Indigenous people are often forced to deny their heritage in order to have access to resources and opportunities that are denied to them as a result of their heritage.	He has argued that the United States has used them to strip indigenous people of their rights, land, and culture in order to benefit those in power. Churchill also argues that these laws were designed to create divisions among indigenous people in order to make them easier to control and manipulate. He has also argued that this manipulation continues today, in that indigenous people are often forced to deny their heritage in order to have access to resources and opportunities that are denied to them because of their heritage.	**5—Approx. equivalent** The paraphrase retains all the information of the original, but uses synonymous language in several places. Also, the paraphrase replaces the pronoun "he" with "Churchill" and vice versa.
40	**When tailoring his 1970 composition "I'm the Greatest" for Starr to record on the 1973 album Ringo, Lennon referenced the song title with the lines** [. . .] Lennon's original line was changed to "Back off Boogaloo" to reference the song's title, creating the line that Starr sang in the finished version.	Lennon's original reference to "I'm the Greatest" was changed to "Back off Boogaloo" to create the line Starr sang on the finished recording.	**4—Nearly equivalent** The original does not specify that the line in question was "I'm the Greatest." However, this can be inferred from the rest of the passage. Otherwise, the paraphrase and the original are quite similar.
40	**Some monuments were vandalized and some were moved to new locations. [. . .]** In some cases, the monuments were moved to new locations. For example, the stairway of the Great Pyramid of Cholula in Mexico was relocated in the 19th century and then returned to its original location in the 20th century. Other monuments were destroyed during wars or religious conflicts, such as the Buddhas of Bamiyan in Afghanistan, which were destroyed by the Taliban in 2001.	In some cases, monuments were moved to new locations. The Great Pyramid of Cholula, in Mexico, was moved in the 19th century and then returned to its original place in the 20th. Others were destroyed during wars or religious controversies, such as the Buddhas of Bamiyan in Afghanistan, which were destroyed by the Taliban in 2001.	**3—Somewhat equivalent** Only one major change in this short text, but I feel it's significant. The original text speaks of the moving of a pyramid's staircase, but the paraphrased text implies that the entire pyramid was moved and then replaced. This massively changes the meaning of this text.
60	**Heaving wildly, the Hunyani began to descend rapidly. [. . .]** The crew was unable to regain control and the Hunyani crashed into the ground at 17:14, killing all onboard. The crash was investigated by the Civil Aviation Authority of Zimbabwe and the investigation concluded that the crash was caused by a combination of pilot error, fatigue, and weather conditions. The pilot had failed to properly assess the weather conditions before takeoff, instead relying on a forecast from a private weather service. The weather conditions worsened as the plane flew and with the pilots already fatigued, they were unable to appropriately react to the changing conditions. The combination of these factors led to the plane losing control and crashing.	The crew could not regain control, and at 17:14 the Hunyani plunged into the ground and all those on board were killed. The accident was investigated by the Civil Aviation Authority of Zimbabwe, which found that the crash was caused by a combination of pilot error, fatigue and the weather. The pilots had not properly examined the weather before take-off, and instead had relied on a forecast from a private meteorological service. The weather deteriorated during the flight, and the exhausted pilots were unable to react properly to the deteriorating conditions.	**5—Approx. equivalent** Well chosen words make the alternate version different but retain the original meaning well. The only difference is that the paraphrased version does not have the concluding sentence, but it does not alter the meaning in any way as the facts have already been included previously.
60	**Sonic's main means of attack is the Spin Attack, in which he curls into a ball and rapidly spins his body, damaging enemies and certain obstacles upon collision. This may be performed by jumping or by rolling on the ground.** Sonic can also use a variety of other moves, such as the Homing Attack, Slide Attack, and Wall Jump. He also has access to power-ups like the Speed Shoes, Invincibility, and the Super Peel Out. In some games, Sonic can also use special items or vehicles to traverse levels.	Also, the character can use a variety of other attacks, such as the homing attack, the slide attack, and the wall jump. In addition, he can acquire power-ups such as the Speed Boost, Super Speed, and the invincibility meter, and in certain games, he may make use of special items or vehicles to traverse the world.	**2—Topically related** In the second part of the paraphrase, the writer ignores the actual moves and skills of Sonic and invents some entirely different ones that not mentioned in the original text. The method of performing the attack moves has also been missed out.

Table 6: The table shows the result of our survey of paraphrase generation papers from 2018 to 2022. We focus on four aspects: (1) whether a model can paraphrase multiple sentences at once, (2) whether a model is able to merge or split an input sentence when appropriate, (3) whether a model takes context surrounding the input sentence into consideration when paraphrasing, and (4) whether a model enables users to control the semantic and syntactic diversity of paraphrases. [1]Granularity levels are *word*, *phrase*, and *sentence*. [2]Meng et al. [2021] use context for their dataset construction, but do not leverage it during training/inference. [3]The diversity score is a combination of the unigram Jaccard distance and the relative position change for unigrams. [4]The code is represented by a three dimensional vector corresponding to semantic similarity as well as syntactic and lexical distances between the input and output sentences.

Paper	Multi-sentence	Merge / Splits	Contextual	Diversity Control
Iyyer et al. [2018]	✗	✗	✗	Constituency parse
Li et al. [2018]	✗	✗	✗	✗
Roy and Grangier [2019]	✗	✗	✗	✗
Witteveen and Andrews [2019]	✓	?	✗	✗
Kumar et al. [2019]	✗	✗	✗	✗
Hu et al. [2019]	✗	✗	✗	Decoding constraints
Chen et al. [2019]	✗	✗	✗	Exemplar
Li et al. [2019]	✗	✗	✗	Granularity control[1]
Goyal and Durrett [2020]	✗	✗	✗	Exemplar
Lewis et al. [2020]	✓	?	✗	✗
Thompson and Post [2020]	✗	✗	✗	n-gram overlap
Kumar et al. [2020]	✗	✗	✗	Exemplar
Kazemnejad et al. [2020]	✗	?	✗	✗
Krishna et al. [2020]	✗	✗	✗	✗
Rajauria [2020]	✗	✗	✗	✗
Meng et al. [2021]	✗	✗	✗[2]	Diversity score[3]
Huang and Chang [2021]	✗	✗	✗	Constituency parse
Lin et al. [2021]	✓	✗	✗	✗
Goutham [2021]	✗	✗	✗	✗
Damodaran [2021]	✗	✗	✗	Binary
Dopierre et al. [2021]	✗	✗	✗	n-gram
Bandel et al. [2022]	✗	✗	✗	Control code[4]
Hosking et al. [2022]	✗	✗	✗	Syntactic sketch
Yang et al. [2022]	✗	✗	✗	Examplar+Keywords
Xie et al. [2022]	✗	✗	✗	✗
DIPPER (ours)	✓	✓	✓	✓

Table 7: Our retrieval defense significantly improves AI-generated text detection accuracy (at 1% FPR) over baselines on all settings, including our most diverse paraphrase attacks (+60L and +60L,60O).

	Open-ended text generation with Wikipedia prompts (300 generated tokens)								
	GPT2-XL			OPT-13B			GPT-3.5 (davinci-003)		
	Original	+ 60L	+ 60L,60O	Original	+ 60L	+ 60L,60O	Original	+ 60L	+ 60L,60O
Baseline methods:									
Watermark	100.0	68.9	57.2	99.9	63.7	52.8	-	-	-
DetectGPT	70.3	8.7	4.6	14.3	0.8	0.3	2.0	0.5	0.0
OpenAI	21.6	13.3	14.8	11.3	9.1	10.0	30.0	15.6	15.6
(Ours) Retrieval over corpus of 3K generations from model itself, with retriever:									
SP	100.0	86.4	81.5	100.0	84.4	77.7	100.0	65.9	49.5
BM25	100.0	99.0	98.0	100.0	97.2	95.3	100.0	58.8	37.4
(Ours) Retrieval over corpus of 9K generations pooled from all three models, with retriever:									
SP	100.0	72.1	63.2	100.0	74.6	65.6	100.0	63.1	45.6
BM25	100.0	85.0	78.7	100.0	87.2	79.1	100.0	58.8	37.4