Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense

Kalpesh Krishna[♠] [*] **Yixiao Song**[♠] **Marzena Karpinska**[♠]
John Wieting[◇] [†] **Mohit Iyyer**[♠] [†]

[♠]University of Massachusetts Amherst, [◇]Google Research

{kalpesh,mkarpinska,miyyer}@cs.umass.edu
yixiaosong@umass.edu jwieting@google.com

Abstract

To detect the deployment of large language models for malicious use cases (e.g., fake content creation or academic plagiarism), several approaches have recently been proposed for identifying AI-generated text via watermarks or statistical irregularities. How robust are these detection algorithms to *paraphrases* of AI-generated text? To stress test these detectors, we first train an 11B parameter paraphrase generation model (DIPPER) that can paraphrase *paragraphs*, optionally leveraging surrounding text (e.g., user-written prompts) as context. DIPPER also uses scalar knobs to control the amount of lexical diversity and reordering in the paraphrases. Paraphrasing text generated by three large language models (including GPT3.5-davinci-003) with DIPPER successfully evades several detectors, including watermarking, GPTZero, DetectGPT, and OpenAI's text classifier. For example, DIPPER drops the detection accuracy of DetectGPT from 70.3% to 4.6% (at a constant false positive rate of 1%), without appreciably modifying the input semantics.

To increase the robustness of AI-generated text detection to paraphrase attacks, we introduce a simple defense that relies on *retrieving* semantically-similar generations and must be maintained by a language model API provider. Given a candidate text, our algorithm searches a database of sequences previously generated by the API, looking for sequences that match the candidate text within a certain threshold. We empirically verify our defense using a database of 15M generations from a fine-tuned T5-XXL model and find that it can detect 80% to 97% of paraphrased generations across different settings, while only classifying 1% of human-written sequences as AI-generated.[1]

[1]Code, data and pretrained paraphrasers will be added to https://github.com/martiansideofthemoon/ai-detection-paraphrases.
*Work partly done as a student researcher in Google Research.
† Equal advising.

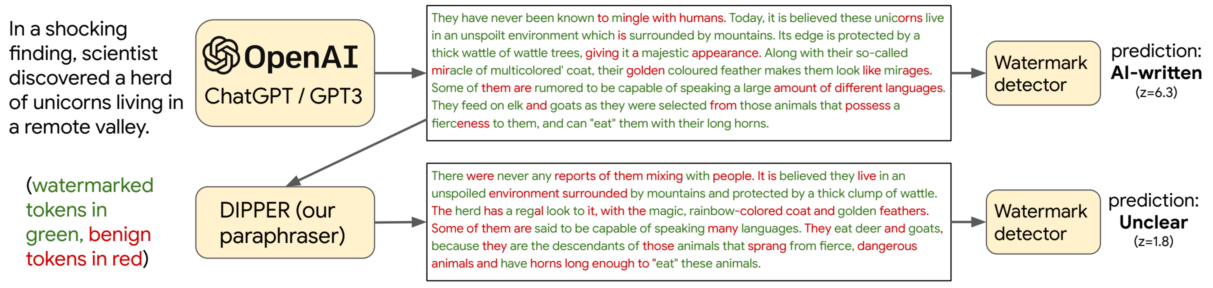
1 Introduction

Large language models (LLMs) such as ChatGPT (Schulman et al., 2022) exhibit an unprecedented ability to write coherent and relevant long-form text in response to user-specified prompts. These abilities have sparked fears of malicious applications such as automatically generating fake news articles or homework answers (Stokel-Walker, 2022). To defend against these use cases, several algorithms have recently been proposed to detect machine-generated text, including watermarking (Kirchenbauer et al., 2023), GPTZero (Tian, 2023), DetectGPT (Mitchell et al., 2023), and OpenAI's text classifier (OpenAI, 2023). However, it remains unclear how robust these algorithms are to *paraphrase attacks*, in which AI-generated text from an LLM is rewritten by another (smaller) model to convey approximately[2] the same meaning but using different word choices and syntax.

In this paper, we first demonstrate the vulnerability of these existing detectors to paraphrase attacks (Section 3, 4). In such attacks, we require an *external* paraphraser model since paraphrases generated by the base LLM are still susceptible to detection techniques such as watermarking. We train an 11B parameter paraphrase generation model called DIPPER[3] to execute these attacks. DIPPER possesses two unique features that help its outputs evade AI-generated text detectors:

1. **Paraphrasing long-form text in context:** Most modern paraphrasers are exclusively trained on sentence-level data, ignoring discourse-level information. However, many critical use cases of LLMs involve generating long-form text in responses to detailed user-specified prompts. Thus, we train DIPPER to paraphrase paragraph-length texts, re-order

[2]We use the quasi-paraphrase definition of semantic equivalence from Bhagat and Hovy (2013) throughout this paper.
[3]DIPPER stands for **Di**scourse **Pa**ra**p**hras**er**.


In a shocking finding, scientist discovered a herd of unicorns living in a remote valley.
OpenAI
ChatGPT / GPT3
(watermarked tokens in green, benign tokens in red)
DIPPER (our paraphraser)
Watermark detector
prediction: AI-written
(z=6.3)
Watermark detector
prediction: Unclear
(z=1.8)

They have never been known to mingle with humans. Today, it is believed these unicorns live in an unspoilt environment which is surrounded by mountains. Its edge is protected by a thick wattle of wattle trees, giving it a majestic appearance. Along with their so-called miracle of multicolored' coat, their golden coloured feather makes them look like mirages. Some of them are rumored to be capable of speaking a large amount of different languages. They feed on elk and goats as they were selected from those animals that possess a fierceness to them, and can "eat" them with their long horns.

There were never any reports of them mixing with people. It is believed they live in an unspoiled environment surrounded by mountains and protected by a thick clump of wattle. The herd has a regal look to it, with the magic, rainbow-colored coat and golden feathers. Some of them are said to be capable of speaking many languages. They eat deer and goats, because they are the descendants of those animals that sprang from fierce, dangerous animals and have horns long enough to "eat" these animals.

Figure 1: An overview of paraphrasing attacks with DIPPER on watermarked text (Kirchenbauer et al., 2023). The original model generation (top) contains several "green" watermarked tokens that are counted by a detector to judge whether the text was written by an AI. After paraphrasing, several of these green tokens are replaced with approximately semantically-equivalent red tokens, thereby fooling the detector (actual outputs from a watermarked version of GPT2-XL and our paraphraser DIPPER shown).

content, and optionally leverage context such as input prompts.

2. **Controlling output diversity:** Another weakness of existing paraphrasers is that they lack an easy way to control output diversity. An attacker may want to apply just the minimum amount of lexical and syntactic modifications necessary to evade a detection algorithm. DIPPER provides users with two intuitive scalar control knobs at inference time that are trained end-to-end: one controls the lexical diversity of the paraphrase, and the other controls the amount of content re-ordering.

We use DIPPER to attack several recently proposed AI-generated text detection algorithms (see Figure 1 for an attack overview). Overall, experiments on multiple LLMs (including GPT3.5-davinci-003) and tasks show that all detection algorithms misclassify a significant amount of AI-generated texts as human-written text after they are paraphrased by DIPPER. As a specific example, DetectGPT (Mitchell et al., 2023) correctly detects 70.3% of model-generated sequences from GPT2-XL, but after paraphrasing, its detection rate drops to only 4.6%[4] despite minimal semantic modification. We confirm the validity of DIPPER's paraphrases through several automatic evaluations as well as a human evaluation of semantic similarity.

Given the vulnerability of AI-generated text detectors to paraphrasing, how can we defend against such attacks? In the second part of our paper (Section 5), we propose to use *retrieval* methods to detect AI-generated text instead of relying on statistical properties of the text or watermarking. First, an LLM API provider stores every output generated by their model in a database. The API provider then offers a service in which a semantic representation of a candidate text is compared to representations of every generation stored in the database. The search focuses on the *semantics* of the input and can leverage both standard IR methods such as BM-25 (Robertson et al., 1995) as well as semantic vector representations such as P-SP from Wieting et al. (2022). Since paraphrasing does not modify the semantics of the input, this algorithm is robust to paraphrasing attacks. Specifically, we find that 97.3% of PG19 paraphrases and 80.4% of Wikipedia paraphrases are successfully detected in a large database of over 15M generations, with a false positive rate of just 1.0%.

In contrast to concurrent work that also uses paraphrasing to attack AI-generated text detectors (Sadasivan et al., 2023), our work offers more comprehensive attack experiments, a new and more powerful paraphraser, human evaluations of paraphrase quality, and finally a novel defense mechanism based on retrieval to combat such attacks. To spur future research in this area, we will release our DIPPER model, its training dataset, and a unified codebase for evaluating both existing detectors and our retrieval-based method.[1]

2 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 genera-

[4]These detection rates were computed at a constant false positive rate of 1%. Due to the importance of low false positive rates in this task, we suggest the community to use a fixed low FPR rather than AUCROC values; more in Section 4.1.

tive algorithm to encode hidden information unique to the API (Section 2.1); (2) statistical outlier detection methods, which do not modify the generative algorithm but look for inherent artifacts in generated text (Section 2.2); and (3) classifiers trained to discriminate machine-generated text from human-written text (Section 2.3). Finally, in Section 2.4, 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.

2.1 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.

2.2 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.

2.3 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.

2.4 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.

3 Building a controllable discourse paraphraser

Having outlined existing methods to detect machine-generated text, we now focus on a simple attack against all detection techniques: *paraphrasing* the generated text. Intuitively, paraphrasing can alter the statistical properties of model generated text thus fooling outlier detection or classification methods. It can also substantially change the number of "green" tokens present in the watermarking approaches (Figure 1). To evade such defenses, a paraphraser must be able to handle *context* in the form of prompts or multi-sentence inputs. Their behavior should also be *controllable* in order to make as many lexical or syntactic modifications as needed to evade a given defense. In all cases, they should not appreciably change the semantics of their inputs. Finally, to evade watermarking, the paraphraser must be implemented with a model that is different from the watermarked model, as otherwise the paraphrased text will also be watermarked. In this section, we detail how we construct a paraphraser (DIPPER) with all of these properties. To gain a better understanding of DIPPER's ability compared to other existing paraphrase models, we survey related work in Appendix A.1.[5]

3.1 Constructing paraphrase data

Our process involves fine-tuning a large language model on a parallel dataset of paragraph-level paraphrases. Our ultimate objective is to develop a model capable of paraphrasing any given sequence of sentences, demarcated by a delimiter and potentially embedded within a broader context, while offering control over the extent of lexical and word-order alterations in the original text. We leverage the PAR3 dataset publicly released by Thai et al. (2022) to train DIPPER. This dataset contains multiple translations of non-English novels into English aligned at a paragraph level (e.g., it contains both the Henry Morley and Robert Adams translations of Voltaire's *Candide*), which we treat as paragraph-level paraphrases and use to train our paraphraser. More formally, let p and q be aligned paragraphs that come from different English translations of the same non-English novel, where $p_1, p_2...p_N$ denote sentences of p and $q_1, q_2, ...q_M$ denote sentences of q. Note that M may not always equal N when two translators disagree on when to merge and split source sentences. Then, we perform the following steps (see Figure 2 for an overview):

1. **Align sentences** of p to sentences of q by using semantic similarity scores from paraphrase simi-

[5]We also note that we observe that DIPPER performs competitively with the much larger and more powerful GPT-3.5 davinci-003 model in terms of paraphrase quality, and significantly better at controlling diversity. This finding shows that specialized smaller models can outperform large language models in paraphrasing tasks. We will provide experiments detailing this comparison in the next version.

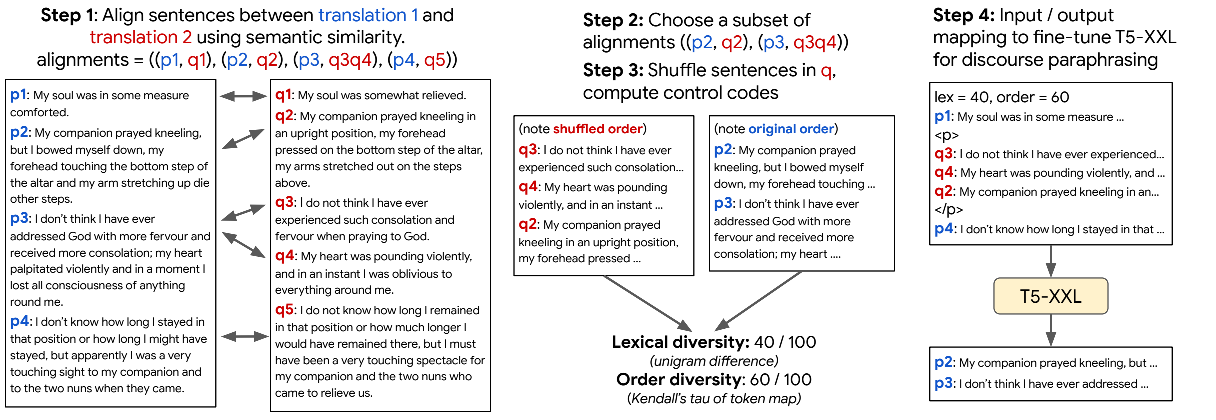

Step 1: Align sentences between translation 1 and translation 2 using semantic similarity.
alignments = ((p1, q1), (p2, q2), (p3, q3q4), (p4, q5))
p1: My soul was in some measure comforted.
p2: My companion prayed kneeling, but I bowed myself down, my forehead touching the bottom step of the altar and my arm stretching up die other steps.
p3: I don't think I have ever addressed God with more fervour and received more consolation; my heart palpitated violently and in a moment I lost all consciousness of anything round me.
p4: I don't know how long I stayed in that position or how long I might have stayed, but apparently I was a very touching sight to my companion and to the two nuns when they came.
q1: My soul was somewhat relieved.
q2: My companion prayed kneeling in an upright position, my forehead pressed on the bottom step of the altar, my arms stretched out on the steps above.
q3: I do not think I have ever experienced such consolation and fervour when praying to God.
q4: My heart was pounding violently, and in an instant I was oblivious to everything around me.
q5: I do not know how long I remained in that position or how much longer I would have remained there, but I must have been a very touching spectacle for my companion and the two nuns who came to relieve us.
Step 2: Choose a subset of alignments ((p2, q2), (p3, q3q4))
Step 3: Shuffle sentences in q, compute control codes
(note shuffled order)
q3: I do not think I have ever experienced such consolation...
q4: My heart was pounding violently, and in an instant ...
q2: My companion prayed kneeling in an upright position, my forehead pressed ...
(note original order)
p2: My companion prayed kneeling, but I bowed myself down, my forehead touching ...
p3: I don't think I have ever addressed God with more fervour and received more consolation; my heart ....
Lexical diversity: 40 / 100
(unigram difference)
Order diversity: 60 / 100
(Kendall's tau of token map)
Step 4: Input / output mapping to fine-tune T5-XXL for discourse paraphrasing
lex = 40, order = 60
p1: My soul was in some measure ...
<p>
q3: I do not think I have ever experienced...
q4: My heart was pounding violently, and ...
q2: My companion prayed kneeling in an...
</p>
p4: I don't know how long I stayed in that ...
T5-XXL
p2: My companion prayed kneeling, but ...
p3: I don't think I have ever addressed ...

Figure 2: An illustration of the method used to train DIPPER on English translations of the French novel *The Nun*. We first align sentences between the two translations to create parallel data. Next, a subset of the alignments are chosen; in this example, we use (p_2, q_2) and (p_3, q_3q_4). We shuffle sentences, compute control codes, and finally fine-tune a T5-XXL LM to generate p_2p_3 given $q_3q_4q_2$ and the context p_1 and p_4.

larity metric in Wieting et al. (2019a) to run the sequence alignment algorithm designed by Needleman and Wunsch (1970) which uses dynamic programming. More details about the semantic similarity metric are discussed in Section 4.1.

2. **Choose a subset of sentences** $p_i...p_j$ from the first paragraph. Let $q_{i'}...q_{j'}$ be the corresponding alignment in the second paragraph. In Figure 2, $i = 2, j = 3, i' = 2, j' = 4$.

3. **Re-order** the sentences $q_{i'}...q_{j'}$ randomly, and compute the **diversity control codes** between $p_i...p_j$ and shuffle$(q_{i'}...q_{j'})$. We shuffle the sentences to allow for the model to learn sentence re-ordering. We compute lexical diversity (L) using unigram token overlap and the order diversity (O) using the Kendall-Tau correlation of tokens of overlapping words between $p_i...p_j$ and shuffle$(q_{i'}...q_{j'})$, which is also used in Krishna et al. (2020). These values are normalized and rounded to a 6-point scale with values $\{0, 20, 40, 60, 80, 100\}$. Here $L = 20$ roughly corresponds to a 20% lexical modification of the input.

4. **Map** the shuffled $q_{i'}...q_{j'}$ to $p_i...p_j$, leveraging context from the rest of p and control codes using string concatenation as follows,

$$\begin{aligned}
\text{codes} &= \text{“lexical} = L, \text{order} = O\text{''} \\
\text{left} &= p_1...p_{i-1} \\
\text{input} &= \texttt{<p>}\ \text{shuffle}(q_{i'}...q_{j'})\ \texttt{</p>} \\
\text{right} &= p_{j+1}...p_N \\
\text{target} &= p_i...p_j \\
\text{codes} \oplus \text{left} &\oplus \text{input} \oplus \text{right} \longrightarrow \text{target}
\end{aligned}$$

where $\oplus$ is the string concatenation operation. During inference time, we can paraphrase an arbitrary sequence of sentences by marking it with <p> tags, assigning values to the control codes L and O depending on the desired diversity.[6]

3.2 Model and training details

Our paraphrase generation model is a sequence-to-sequence Transformer neural network (Vaswani et al., 2017). We initialize our model with the T5-XXL checkpoint (Raffel et al., 2020) and fine-tune it on our paraphrase generation data, using early stopping on validation loss for held-out novels. We found it helpful to paraphrase a maximum of 3 consecutive sentences at time, as it led 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 for T5-XXL v1.1. At inference time, we sample from our model using nucleus sampling (Holtzman et al., 2020) with $p = 0.75$ and a variety of control codes.

4 Experiments attacking AI-detection algorithms with DIPPER

In this section, we describe experiments attacking existing AI-generated text detectors with our DIPPER paraphraser. We first detail our experimental setup (metrics, models, and detectors) in

[6]To make our paper presentation more intuitive, we have slightly modified the notation that our actual pretrained paraphraser 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 API and code release.

Section 4.1 and 4.2 before moving on to discuss our results (Section 4.3). Overall, we find that paraphrasing easily evades all detectors across three language models (GPT2-XL, OPT-13B, GPT3.5-davinci-003) and two tasks (open-ended generation and long-form question answering).

4.1 Evaluation metrics

We use two metrics to automatically evaluate attack success rates after paraphrasing: (1) detection accuracy, and (2) semantic similarity between the original LM generations and DIPPER paraphrases.[7]

Detection accuracy: Our first metric measures how often the input text is correctly detected as AI-generated. Since detection rates are heavily dependent on the chosen detection threshold, the AUC-ROC metric is commonly used to measure detector performance (Mitchell et al., 2023), which considers the range of all possible thresholds. However, in this application, it is critical that the *false positive rate* is low; in other words, human-written text must almost never be classified as machine-generated (OpenAI, 2023; Kirchenbauer et al., 2023). Hence, we fix the false positive rate to 1% for all detection algorithms (although even 1% is likely too high in practice), and adjust the detection threshold accordingly while reporting detection accuracies. Additionally, we also plot ROC curves focusing on the region between FPR=0% and FPR=1%. Overall, at a constant false positive rate of 1%, we expect detection rates to plummet on paraphrased text.

Semantic similarity (Sim): Detection accuracy is an insufficient evaluation of our attack's success. We also need to measure whether the original and paraphrased generations share approximately the same semantics. We measure semantic similarity using the P-SP model from Wieting et al. (2022), which is a simple embedding averaging model trained on a large corpus of filtered paraphrase data (Wieting and Gimpel, 2018).[8] Despite its simplicity, P-SP achieves state-of-the-art performance on unsupervised semantic similarity tasks, while being four orders of magnitude faster on CPU than methods using pretrained Transformers. We consider semantics to be approximately preserved if the P-SP score is higher than 0.76, which is the median P-SP score of human-written paraphrase pairs on the PAR3 dataset (Thai et al., 2022). Finally, since automatic metrics for text generation are limited (Celikyilmaz et al., 2020), we additionally conduct a human evaluation of DIPPER's paraphrase quality in Section 6.2.

4.2 Models, datasets & detection algorithms

Base language models: We conduct attacks on three language models of varying sizes that belong to different model families. We consider the GPT2-XL model which has 1.5B parameters (Radford et al., 2019), the OPT-13B model (Zhang et al., 2022), and the `text-davinci-003` variant from the GPT-3.5 family of models (Brown et al., 2020), which has 175B parameters and has additionally been instruction tuned using reinforcement learning from human feedback (Ouyang et al., 2022). For all language models, we sample generations that are 300 tokens long before passing them through DIPPER for the attack experiments.[9]

Natural language generation tasks: We experiment with two text generation tasks whose outputs are long-form text since most malicious applications (e.g., fake article creation) are associated with long-form outputs. First, we consider *open-ended generation*, in which an LM must generate a continuation to a passage given a two-sentence prompt. To obtain our prompts, we sample 3K contiguous two-sentence chunks from the validation split of the WikiText-103 dataset (Merity et al., 2017), and use the next 300 tokens as the "human-written" continuation. Second, we 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.[10] 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

[7] Additional aspects of paraphrase quality like continuity to prompt, fluency, and coherence are evaluated in Section 6 using GPT-3.5 perplexity (Brown et al., 2020) and RankGen (Krishna et al., 2022a).

[8] This is an improved version, trained on more data, from models proposed in prior work (Wieting et al., 2019a,b).

[9] For GPT2-XL and OPT-13B, we generate text using nucleus sampling (Holtzman et al., 2020) with $p = 0.9$. For `davinci-003`, we use the default hyperparameter settings on the API Playground of temperature = 0.7 to sample tokens.

[10] 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.

Paraphrase attacks on open-ended generation (Wikipedia prompts, 300 generated tokens)						
Metric →	Sim ↑	Detection Accuracy ↓				
Detector →		Watermarks (2023)	DetectGPT (2023)	OpenAI (2023)	GPTZero (2023)	RankGen (2022a)
GPT2-1.5B	-	100.0	70.3	21.6	13.9	**13.5**
+ DIPPER 20L	99.2	97.1	28.7	19.2	9.1	15.8
+ DIPPER 40L	98.4	85.8	15.4	17.8	7.3	18.0
+ DIPPER 60L	96.9	68.9	8.7	**13.3**	7.1	19.8
+ DIPPER 60L, 60O	94.3	**57.2**	**4.6**	14.8	**1.2**	28.5
OPT-13B	-	99.9	14.3	11.3	8.7	**3.2**
+ DIPPER 20L	99.1	96.2	3.3	11.8	5.4	5.2
+ DIPPER 40L	98.6	84.8	1.2	11.6	3.8	6.6
+ DIPPER 60L	97.1	63.7	0.8	**9.1**	6.3	9.3
+ DIPPER 60L, 60O	94.6	**52.8**	**0.3**	10.0	**1.0**	13.5
GPT-3.5-175B, davinci-003	-	-	26.5*	30.0	7.1	**1.2**
+ DIPPER 20L	97.6	-	12.5*	20.6	4.3	1.7
+ DIPPER 40L	96.7	-	8.0*	22.4	4.8	2.0
+ DIPPER 60L	94.2	-	7.0*	**15.6**	6.1	3.9
+ DIPPER 60L, 60O	88.4	-	**4.5***	**15.6**	**1.8**	7.3
Human Text	-	1.0	1.0	1.0	1.0	1.0

Table 1: Performance of various AI detection algorithms (at 1% false positive rate or FPR) before and after DIPPER paraphrasing on open-ended generation using Wikipedia prompts. As the lexical diversity (L) and re-ordering (O) increases, detection accuracy decreases across algorithms, with nearly perfect semantic similarity (Sim). *GPT3.5 DetectGPT scores computed using 200 samples at a 20% FPR since it scores 0% at a 1% FPR.

its longest human-written answer, which yields 3K long-form QA pairs. Note that in both tasks, the human references (continuation for the first task and answer for the second) are only used to adjust the detection thresholds of our studied detectors to maintain a 1% FPR.[11]

AI-text detection algorithms: Our main experiments use five AI detection algorithms: (1) watermarking (Section 2.1); (2) DetectGPT (Section 2.2); (3) GPTZero (Section 2.2); (4) OpenAI's text classifier (Section 2.3); and (5) RankGen-XL-all (Krishna et al., 2022a).[12] We respect the minimum length specifications of each detector, discarding instances where any of the machine-generated text, human-written text, or paraphrased text is shorter than the minimum length. We use the default hyperparameters for each detection algorithm.

Paraphrasing AI-generated text: We pass the prompts for each task and AI-generated responses to those prompts through DIPPER. Specifically, we feed the model input of the form

lex = L, order = O prompt <p> generated-text </p>,

where L and O represent the paraphraser diversity control codes and the <p> and </p> special tokens mark the boundaries of the text to be paraphrased. We use $L = 20, 40, 60$ and $O = 0, 60$ in our main attack experiments. After paraphrasing, we ensure that the machine-generated sequence, paraphrased sequence, and human-written sequence have an equal number of words by truncating them to the length of the shortest among the three. To ensure higher semantic preservation, we iteratively paraphrase long sequences three sentences at a time, keeping already paraphrased text in the context of the generation. In our experiments, we only **paraphrase each generation once**, while a sophisticated attacker may sample multiple paraphrases and use the sample that best evades detection while preserving semantics; we discuss this further in Section 4.4.

4.3 Attacking AI-generated text detectors

We present our attack results in Table 1 for open-ended generation and in Table 2 for long-form question answering. Overall, we observe the following:

Paraphrasing significantly lowers detection accuracy while preserving input semantics. Across all language models, detectors (except for

[11]Evaluating how well the studied LMs perform on these two tasks is a challenging problem in its own right that could make additional use of the human references, but this is irrelevant to our paper.

[12]While RankGen was not explicitly optimized for AI text detection, it was trained to identify well-written continuations to a prompt to improve generation quality, so we hypothesize that it could also act as a reasonable AI-generated text detector.

Long-form Question Answering, 300 generated words				
Metric →	Sim ↑	Detection Accuracy ↓		
		W.Mark	D.GPT	OpenAI
GPT2-1.5B	-	100.0	74.9	59.2
+ DIPPER 20L	99.5	98.9	45.7	35.3
+ DIPPER 40L	99.0	90.7	28.0	34.4
+ DIPPER 60L	97.5	71.1	15.8	**31.3**
+ 60L, 60O	96.2	**55.8**	**7.6**	32.7
OPT-13B	-	100.0	29.8	33.5
+ DIPPER 20L	99.6	98.3	15.0	24.5
+ DIPPER 40L	99.4	87.3	6.4	24.1
+ DIPPER 60L	96.5	65.5	3.2	**21.6**
+ 60L, 60O	92.9	**51.4**	**1.5**	**21.6**
GPT-3.5-175B davinci-003	-	-	67.0*	40.5
+ DIPPER 20L	99.9	-	54.0*	43.1
+ DIPPER 40L	99.8	-	36.0*	43.1
+ DIPPER 60L	99.5	-	23.0*	40.1
+ 60L, 60O	98.3	-	**14.0***	**38.1**
Human Text	-	1.0	1.0	1.0

Table 2: Performance of AI-text detection algorithms (at 1% false positive rate) before and after DIPPER paraphrasing on long-form question answering. As the lexical (L) and order (O) diversity increases, detection accuracy decreases across algorithms, with nearly perfect semantic similarity (Sim). *GPT3.5 DetectGPT scores computed using 100 samples at a 20% FPR, since it scores 0% at a 1% FPR.

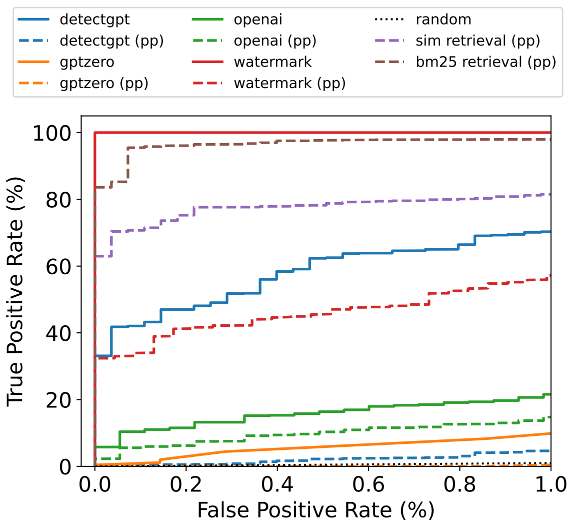


Figure 3: ROC curves (with X-axis clipped at 1% false positive rate or FPR) for GPT2-XL using different AI-detection algorithms. Curves are shown before (solid lines) and after paraphrasing (dashed). Paraphrasing reduces detection accuracy across FPRs. Our proposed defense (retrieval) has the highest accuracy after paraphrasing. See Appendix C for unclipped ROC curves.

RankGen[13]), and tasks, paraphrasing significantly lowers detection accuracy across all diversity control codes. For instance, paraphrasing GPT2-XL open-ended generations reduces watermark detection accuracy from 100% to 57.2%, and DetectGPT accuracy from 70.3% to just 4.6%. Trends are similar even for large language models like GPT-3.5, for which paraphrasing reduces OpenAI's text classifier accuracy from 30.0% to 15.6%. Additionally, DIPPER preserves semantics effectively, as 88.4%-99.9% paraphrases achieve a P-SP score higher than the median P-SP score of human-written paraphrases in PAR3. Among the different detection algorithms, we find that watermarking is the most resilient to paraphrasing.

Non-watermarking detectors are generally ineffective. We observe that all AI-generated text detectors apart from watermarking struggle with text generated by larger models like OPT-13B and GPT-3.5 davinci-003, achieving detection accuracies lower than 50%. For example, while DetectGPT is effective on the smaller GPT2-XL model with a detection accuracy of 74.9% on long-form QA, its accuracy drops to just 29.8% on OPT-13B. Furthermore, GPTZero and RankGen perform the worst among the five detectors on all tested language models (Table 1), as they are only able to detect fewer than 15% of non-paraphrased AI-generated text in all settings. Thus, we recommend against using these models for detecting AI-generated text.

ROC plots confirm the trends at different false positive rates. In Figure 3, we plot the detection accuracy (true positive rate) at different values of false positive rates between 0% and 1% for GPT2-XL. Overall, paraphrasing significantly drops detection rates across all false positive rate thresholds. ROC plots for GPT2-XL at larger false positive rates are available in Appendix C.

4.4 Alternative paraphrasing attacks

Here, we discuss two other (untested) ways to attack AI-generated text detectors via paraphrasing, which further showcase the brittleness of existing detectors.

Paraphrasing multiple times: Our presented attacks use just a single paraphrase generated by DIPPER to evade detection. An simple way to further improve the effectiveness of a paraphrase attack

[13]RankGen scores paraphrased text as AI-generated more often than non-paraphrased text. We attribute this to paraphrases being poorer continuations to the prompt compared to the original (Section 6), an aspect RankGen bases its score on. Unfortunately, low overall detection rates with or without paraphrasing make it a poor AI-generated text detector.

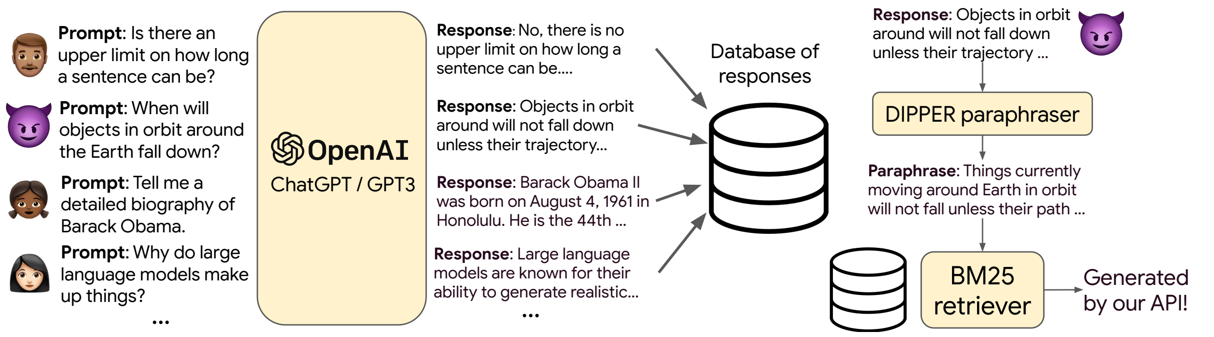

Prompt: Is there an upper limit on how long a sentence can be?
Prompt: When will objects in orbit around the Earth fall down?
Prompt: Tell me a detailed biography of Barack Obama.
Prompt: Why do large language models make up things?
...
OpenAI
ChatGPT / GPT3
Response: No, there is no upper limit on how long a sentence can be....
Response: Objects in orbit around will not fall down unless their trajectory...
Response: Barack Obama II was born on August 4, 1961 in Honolulu. He is the 44th ...
Response: Large language models are known for their ability to generate realistic...
...
Database of responses
Response: Objects in orbit around will not fall down unless their trajectory ...
DIPPER paraphraser
Paraphrase: Things currently moving around Earth in orbit will not fall unless their path ...
BM25 retriever
Generated by our API!

Figure 4: An illustration of AI-generated text detection with retrieval. Several users (including the attacker, shown as the purple emoji) feed prompts to the API which are collectively added to a private API-side database. Candidate queries are compared against this database using a retriever like BM25 or P-SP. Empirically, we find that this defense is quite effective at detecting paraphrases from an attacker (as shown in the figure).

is to sample multiple times[14] from DIPPER and choose a paraphrase that evades the detector while also preserving semantics. We do not perform this attack since it can only be done if the attacker has access to a detector, which may be a strong assumption (more in Section 5.5). That said, we believe leveraging multiple paraphrase samples can make these kinds of paraphrase attacks even more potent against publicly available detectors.

Non-DIPPER paraphrasers: A second alternative is to use non-DIPPER paraphrasers that operate at the sentence level. These models can be deployed for long-form text inputs by paraphrasing the inputs sentence by sentence, ignoring prompt context. While the concurrent work of Sadasivan et al. (2023) shows that this method can also evade detection, our ablations in Section 6 show that these paraphrasers have lower quality and are less compatible with the prompt as DIPPER paraphrasers. A more interesting option is to use a large language model served by an API to perform few-shot contextual paraphrasing. While this method is likely to provide accurate paraphrases, they may be detectable by strategies like watermarking (whether using the same API provider as the original LLM or a different provider). We thus expect a sophisticated adversary to use their own private paraphraser to evade detection (DIPPER in our simulated attacks).

5 Defense against paraphrase attacks using retrieval

In Section 4.3, we observed that paraphrasing is an effective attack against AI-generated text detection algorithms. How can API providers who serve outputs from large language models (LLMs) defend against these attacks? In this section, we propose *retrieval* over previously-generated sequences as a defense against paraphrase attacks. At a high level (Figure 4), an API provider first stores every sequence generated by their LLM in a database. The API provider offers an interface that allows users to enter candidate AI-generated text as a query. The interface searches over the entire database of previously-generated text, trying to find a sequence that approximately matches the content of the input query. This search can be done using a semantic similarity scorer like SP (Wieting et al., 2022) or a retriever like BM25 (Robertson et al., 1995). Since paraphrasing approximately preserves input semantics, we expect such a defense to still be able to map paraphrased generations to their source.

In this section we first formalize our retrieval defense in Section 5.1. We then perform controlled comparisons of retrieval with other detection algorithms (Section 5.2) and evaluate our method at scale using a large retrieval corpus of 15M generations (Section 5.3).

5.1 Formulating the retrieval defense

Let f_{LM} be an LLM API (e.g., GPT-3.5) that takes a prompt x as input and returns a continuation y. Let f_{ret} be an encoder (e.g., TF-IDF, neural network) that embeds variable-length sequences into fixed-size vectors that represent the input semantics. Then, we do the following:

1. **Building the database**: Let $x_1, ..., x_N$ be the set of prompts that have been fed as input to the API in the past, where N can potentially be very large for popular APIs (we study up to $N = 15$M).

[14]Precisely, compute $f_{\text{dipper}}(x)$ for different random seeds during nucleus or temperature sampling. Alternatively, an attacker could also compute $f_{\text{dipper}}(f_{\text{dipper}}(...f_{\text{dipper}}(x)))$, but this will lead to excessive semantic drift from x.

Long-form Question Answering (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	71.1	55.8	100.0	65.5	51.4	-	-	-
DetectGPT	74.9	15.8	7.6	29.8	3.2	1.5	1.0	0.0	0.0
OpenAI	59.2	31.3	32.7	33.5	21.6	21.6	40.5	40.1	38.1
(Ours) Retrieval over corpus of 3K generations from model itself, with retriever:									
SP	100.0	95.6	87.7	100.0	94.8	85.3	100.0	94.2	85.1
BM25	100.0	99.2	97.8	100.0	99.3	97.3	100.0	98.6	96.2
(Ours) Retrieval over corpus of 9K generations pooled from all three models, with retriever:									
SP	100.0	88.9	75.4	100.0	89.6	76.4	100.0	93.8	84.6
BM25	100.0	98.3	95.2	100.0	98.5	94.4	100.0	98.5	96.0
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

Table 3: Our proposed defense (retrieval) significantly improves AI-generated text detection accuracy (at false positive rate 1%) over baselines on all settings, including our most diverse paraphrase attacks (+60L, +60L,60O).

We construct our database $\mathbf{Y}$ by:

$$\begin{aligned} y_i &= f_{\text{LM}}(x_i) \quad \text{for } i = 1, ..., N \\ \mathbf{y}_i &= f_{\text{ret}}(y_i) \quad \text{for } i = 1, ..., N \\ \mathbf{Y} &= [\mathbf{y}_1, ...\mathbf{y}_N] \end{aligned}$$

The database $\mathbf{Y}$ is dynamically updated and stored on the API side. It is inaccessible to clients except via the API described in the next step.

2. **Querying the database**: Let y' be a candidate text, and suppose a client wishes to know whether y' was generated by the API f_{LM}. The API provider can check this by computing whether the maximum similarity score of y' to an entry in the database exceeds some threshold:

$$\begin{aligned} \mathbf{y}' &= f_{\text{ret}}(y') \\ \text{score} &= \max_i \frac{\mathbf{y}' \cdot \mathbf{y}_i}{|\mathbf{y}'|\,|\mathbf{y}_i|} \\ \text{output} &= \text{score} > L \end{aligned}$$

Here, L is the detection threshold chosen by the API provider. We expect unperturbed machine-generated text to always get a score of 1.0, while paraphrasing the text will lower the detection score. Hence, lowering L will increase the detection rate of heavily-paraphrased text but also increase the false positive rate (i.e., human-written text that resembles sequences previously generated by the LLM API can be falsely flagged). Since N can be very large, the score can also be approximated using efficient nearest neighbor libraries like FAISS (Johnson et al., 2019). However, in this work we only compute exact inner products.

Choice of retriever f_{ret}: We experiment with two choices for f_{ret} including P-SP from Wieting et al. (2022) and BM25 (Robertson et al., 1995). We implement BM25 using the `retriv` library from Bassani (2022). In order to normalize and calibrate BM25 scores, we compute the unigram token overlap (using the evaluation script from Rajpurkar

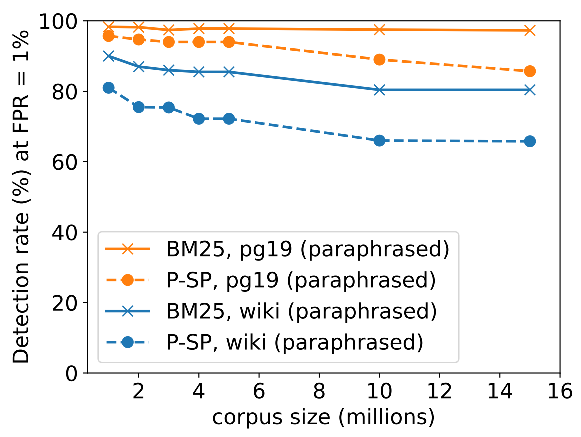


Figure 5: The effectiveness of retrieval as a detection strategy as a function of retrieval corpus size. In all settings we note high detection accuracy of paraphrases (at 1% FPR), which only slightly degraded as the corpus is scaled from 1M to 15M generations.

et al., 2016) between the candidate y' and the best retrieval $y*$ to get a detection score in $[0, 1]$.

5.2 Controlled comparisons of retrieval with other AI-generated text detectors

First, we conduct a controlled comparison between the detection algorithms evaluated in Section 4.3 and our retrieval method. We construct two retrieval corpora for this experiment: (a) a corpus of the 3K sequences generated by a specific LM for one of the tasks; and (b) a corpus of 9K sequences formed by concatenating the generations from all three LMs considered in this paper. We expect (b) to be a more difficult test for our method than (a), since the retriever needs to distinguish between multiple generations from different models given the same prompt. Next, we perform retrieval over this corpus using the original AI-generated text, its paraphrase (generated by DIPPER using control codes L60 and L60,O60), and human-written text as queries. While in this experiment, we only consider queries that are at least 50 token long, we discuss the effect of input length in Section 5.3.

Table 3 shows that **across all LMs, retrieval is a much more effective detector than baseline detectors**. On unperturbed machine-generated text, retrieval has a 100% detection accuracy due to exact match with the retrieval corpus. On paraphrased text, retrieval with BM25 is quite effective, detecting 97.8% of the highest-diversity paraphrases (L60,O60) on GPT2-XL, 97.3% on OPT-13B and 96.2% on GPT-3.5 in long-form question answering. This is significantly better than the next best alternative with competing detectors (55.8%, 51.4%, 38.1%, respectively). Even on our harder augmented database of 9K generations, detection rates continue to be high (95.2%, 94.4%, 96.0%). Finally, we observe that BM25 is a more effective retriever than P-SP, scoring 95.2% vs 75.4% on the augmented setting in GPT2-XL with long-form question answering. These trends are consistent across different false positive rate thresholds, as shown in our ROC curves (Figure 3).

5.3 Is retrieval an effective detector with a large retrieval corpus?

In the previous section, we conducted experiments using the set of 9K sequences generated by all three models as the retrieval corpus. However, this is more of a toy experiment: in practice, a popular LLM API may serve millions of queries a day. As the corpus grows larger, the false positive rate (i.e., human-written text falsely detected as machine-generated) will grow. How well do retrieval-based detectors scale? To answer this question, we need access to a large corpus of machine-generated text. We utilize the training data used to train RankGen (Krishna et al., 2022a), which contains over 70M machine-generated sequences. We use the Project Gutenberg and Wikipedia splits of the training data, each of which contain 15M sequences generated by a T5-XXL model (Raffel et al., 2020) fine-tuned on the different documents in the same domain. We discard generations which are shorter than 50 tokens, and paraphrase a small subset of 2K generations to evaluate retrieval.

Retrieval is effective even with a corpus size of 15M generations. In Figure 5, we plot the detection accuracy as a function of retrieval database size. Overall, we observe that detection accuracy remains consistently high across different corpus sizes (varying from 1M generations to 15M generations). We do observe slight drops in performance as the corpus size increases: just 1% (98.3 to 97.3) on Project Gutenberg (PG19) and 9.6% (90.0 to 80.4) on Wikipedia. Consistent with the results in Section 5.2, BM25 continues to outperform P-SP across different retrieval corpus sizes.

Retrieval detection works best with 50 or more tokens of generated text. Another important factor for our retrieval-based detector is the query length: shorter queries are likely to have more matches (many of them spurious) compared to

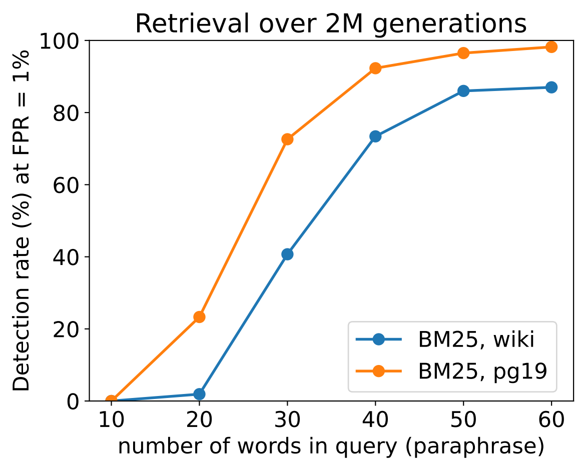


Figure 6: The variation in retrieval performance as a detector with different query lengths. Overall, retrieval performs best with queries of length 50 tokens or more.

longer ones. In Figure 6, we plot the detection accuracy of paraphrased sequences at various query lengths by truncating each sequence to its first X words before using it as a query for BM25. We use a retrieval corpus of 2M generations for this experiment. We observe that BM25 struggles to detect paraphrased text with a query length of 20 (less than 25% accuracy), but the detection rate rapidly increases and begins to plateau at 50 tokens.

5.4 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.

5.5 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[15] and Bard[16]; (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.

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.

6 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 (Section 6.1) and human evaluations (Section 6.2) 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.

6.1 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 A.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

[15] `https://chat.openai.com`
[16] `https://bard.google.com`

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% $_{39.4}$

Table 4: 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).

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 3.1) 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 4 (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).

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 4 (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 gener-

L	Original	Paraphrase	Annotator Comment
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.
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 5: Representative model outputs of each lexical code (L), along with Likert ratings and justifications from the annotators (native English teachers and editors). The bolded text in the original paragraphs are the context. Red words represent the content being changed in the original text (manually marked by the authors), and green words show the corresponding text in DIPPER's paraphrases.

ations (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 4 (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).

6.2 Human evaluation of semantic preservation using DIPPER

The automatic semantic similarity scores in Table 1 and 2 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[17] to evaluate the semantic fidelity of the paraphrases. As human evaluation is expensive, we 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. For each code, we randomly select 20 original and paraphrased text pairs for our annotators to evaluate semantic similarity on a five-point Likert scale (see Appendix B for more details

[17] https://www.upwork.com

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%

Table 6: 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.

of the scale and interface).[18] The rate at which each point on the scale is chosen is reported in Table 6. Annotators also must provide free-form comments justifying their ratings. 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). Details about inter-annotator agreement are in Appendix B.

A qualitative analysis of the free-form annotator comments reveals systemic strengths and shortcomings of DIPPER. We provide a brief analysis below and include three representative examples in Table 5. A comprehensive discussion with more examples can be found in Appendix B.

Strengths: The first case in Table 5 exemplifies DIPPER's ability to leverage information from context to increase diversity while maintaining coherence. No prior paraphraser (see Table 7 for a list) is capable of doing so. The paraphrase in the second row highlights DIPPER's ability to make significant changes to original texts with a high lexical diversity code while preserving their semantic meaning.

Shortcomings: One shortcoming of DIPPER is that, when the original text contains unique proper nouns, such as *Homing Attack* and *Slide Attack* in row 3, applying DIPPER with a high lexical code can modify such nouns. This causes an undesirable semantic drift, as the original proper nouns must be maintained in any valid paraphrase. However, this shortcoming can be mitigated by decreasing the lexical code as shown in Appendix B.

7 Conclusion

We present DIPPER, a discourse paraphrase generation model that can rewrite multiple sentences of text and optionally leverage surrounding context. We use DIPPER to stress test current AI-generated text detectors, and we find that DIPPER paraphrases easily evade these detectors while approximately preserving input semantics. To defend against such paraphrase attacks, we propose a simple retrieval-based mechanism in which we search through a corpus of previously-generated sequences from an LLM API for semantically-similar generations to a given query. Our experiments show that this retrieval defense significantly outperforms baseline detectors on paraphrased text, and is effective at scale. We extensively discuss possible limitations of our defense, and we open source our pretrained models, code, and data to enable the research community to build on these ideas.

[18]The original texts are preceded by their context. The annotators see the same amount of text as DIPPER.

Acknowledgements

We are very grateful to Kenton Lee, Katherine Thai, Slav Petrov, Fernando Pereira, Jon Clark, William Cohen, Tu Vu, Yapei Chang, Shufan Wang and the UMass NLP group for several useful discussions on earlier versions of this paper. Kalpesh Krishna was supported by a Google PhD Fellowship. He also did a part of this work as a Student Researcher in Google Research, hosted by John Wieting.

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 (Section 5.4), as well as discussing possible limitations and approaches to tackling them (Section 5.5). We hope that retrieval-based AI-generated text detectors rapidly improve and are eventually deployed in conjunction with other detection methods.

References

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 121–140.

Ruchit Rajeshkumar Agrawal, Marco Turchi, and Matteo Negri. 2018. Contextual handling in neural machine translation: Look behind, ahead and on both sides. In *21st annual conference of the European association for machine translation*, pages 11–20.

Sameer Badaskar, Sachin Agarwal, and Shilpa Arora. 2008. Identifying real or fake articles: Towards better language modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*.

Elron Bandel, Ranit Aharonov, Michal Shmueli-Scheuer, Ilya Shnayderman, Noam Slonim, and Liat Ein-Dor. 2022. Quality controlled paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 596–609, Dublin, Ireland. Association for Computational Linguistics.

Elias Bassani. 2022. retriv: A user-friendly and efficient search engine in python.

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

Rahul Bhagat and Eduard Hovy. 2013. Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the Association for Computational Linguistics*.

Michael Connor and Dan Roth. 2007. Context sensitive paraphrasing with a global unsupervised classifier. In *European Conference on Machine Learning*, pages 104–115. Springer.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. In *International Conference on Learning Representations*.

Ashwin Devaraj, Iain Marshall, Byron Wallace, and Junyi Jessy Li. 2021. Paragraph-level simplification of medical texts. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4972–4984.

Thomas Dopierre, Christophe Gravier, and Wilfried Logerais. 2021. PROTAUGMENT: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2454–2466, Online. Association for Computational Linguistics.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567.

Sonal Garg, Sumanth Prabhu, Hemant Misra, and G Srinivasaraghavan. 2021. Unsupervised contextual paraphrase generation using lexical control and reinforcement learning. *arXiv preprint arXiv:2103.12777*.

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

Ramsri Goutham. 2021. High-quality sentence paraphraser using transformers in nlp.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. *Proceedings of the Association for Computational Linguistics*.

EA Grechnikov, GG Gusev, AA Kustarev, and AM Raigorodsky. 2009. Detection of artificial texts. *RCDL2009 Proceedings. Petrozavodsk*, pages 306–308.

Alexei Grinbaum and Laurynas Adomaitis. 2022. The ethical need for watermarks in machine-generated language. *arXiv preprint arXiv:2209.03118*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proceedings of the International Conference on Learning Representations*.

Tom Hosking, Hao Tang, and Mirella Lapata. 2022. Hierarchical sketch induction for paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2489–2501, Dublin, Ireland. Association for Computational Linguistics.

Juliane House. 2006. Text and context in translation. *Journal of pragmatics*, 38(3):338–358.

Dirk Hovy. 2016. The enemy in your own camp: How well can we detect statistically-generated fake reviews – an adversarial study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

J Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6521–6528.

Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1022–1033, Online. Association for Computational Linguistics.

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.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Sébastien Jean, Ankur Bapna, and Orhan Firat. 2019. Fill in the blanks: Imputing missing sentences for larger-context neural machine translation. *arXiv preprint arXiv:1910.14075*.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Amirhossein Kazemnejad, Mohammadreza Salehi, and Mahdieh Soleymani Baghshah. 2020. Paraphrase generation by learning how to edit from samples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022a. RankGen: Improving text generation with large ranking models. *arXiv preprint arXiv:2205.09726*.

Kalpesh Krishna, Deepak Nathani, Xavier Garcia, Bidisha Samanta, and Partha Talukdar. 2022b. Few-shot controllable style transfer for low-resource multilingual settings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7439–7468.

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.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2018. Modeling coherence for neural machine translation with dynamic and topic caches. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph text. In *Association for Computational Linguistics*.

Thomas Lavergne, Tanguy Urvoy, and François Yvon. 2008. Detecting fake content with relative entropy scoring. *PAN*, 8:27–31.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33:18470–18481.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

Zhe Lin, Yitao Cai, and Xiaojun Wan. 2021. Towards document-level paraphrase generation with sentence rewriting and reordering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Elman Mansimov, Gábor Melis, and Lei Yu. 2021. Capturing document context inside sentence-level neural machine translation models with self-training. In *Proceedings of the 2nd Workshop on Computational Approaches to Discourse*, pages 143–153.

Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. 2021. A survey on document-level neural machine translation: Methods and evaluation. *ACM Computing Surveys (CSUR)*, 54(2):1–36.

Aurélien Max. 2009. Sub-sentencial paraphrasing by contextual pivot translation. In *Proceedings of the 2009 Workshop on Applied Textual Inference (TextInfer)*, pages 18–26.

Yuxian Meng, Xiang Ao, Qing He, Xiaofei Sun, Qinghong Han, Fei Wu, Chun Fan, and Jiwei Li. 2021. ConRPG: Paraphrase generation using contexts as regularizer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2551–2562.

Hasan Mesut Meral, Bülent Sankur, A Sumru Özsoy, Tunga Güngör, and Emre Sevinç. 2009. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23(1):107–125.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Empirical Methods in Natural Language Processing*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

OpenAI. 2022. OpenAI Models - GPT3.5.

OpenAI. 2023. AI text classifier.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Andrei Popescu-Belis, Sharid Loáiciga, Christian Hardmeier, and Deyi Xiong, editors. 2019. *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*. Association for Computational Linguistics, Hong Kong, China.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Arpit Rajauria. 2020. Pegasus fine-tuned for paraphrasing.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, et al. 2022. Scaling up models and data with T5X and seqio. *arXiv preprint arXiv:2203.17189*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6033–6039, Florence, Italy. Association for Computational Linguistics.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Uribe, Liam Fedus, Luke Metz, et al. 2022. ChatGPT: Optimizing language models for dialogue.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Chris Stokel-Walker. 2022. Ai bot chatgpt writes smart essays-should academics worry? *Nature*.

Katherine Thai, Marzena Karpinska, Kalpesh Krishna, William Ray, Moira Inghilleri, John Wieting, and Mohit Iyyer. 2022. Exploring document-level literary machine translation with parallel paragraphs from world literature. In *Empirical Methods in Natural Language Processing*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Brian Thompson and Matt Post. 2020. Paraphrase generation as zero-shot multilingual translation: Disentangling semantic similarity from lexical and syntactic diversity. In *Proceedings of the Fifth Conference on Machine Translation*, pages 561–570, Online. Association for Computational Linguistics.

Edward Tian. 2023. Gptzero: An ai text detector.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 82–92, Copenhagen, Denmark.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. 2020-2022. Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. 2005. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5998–6008.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. Context-aware monolingual repair for neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 877–886.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019b. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. In *Proceedings of the Association for Computational Linguistics*.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2826–2831.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019a. Beyond BLEU:training neural machine translation with semantic similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4344–4355, Florence, Italy. Association for Computational Linguistics.

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.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019b. Simple and effective paraphrastic similarity from parallel translations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-kirkpatrick. 2022. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 379–388, Abu Dhabi, UAE. Association for Computational Linguistics.

Sam Witteveen and Martin Andrews. 2019. Paraphrasing with large language models. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 215–220, Hong Kong. Association for Computational Linguistics.

Xuhang Xie, Xuesong Lu, and Bei Chen. 2022. Multi-task learning for paraphrase generation with keyword and part-of-speech reconstruction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1234–1243, Dublin, Ireland. Association for Computational Linguistics.

Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Modeling coherence for discourse neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7338–7345.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Kexin Yang, Dayiheng Liu, Wenqiang Lei, Baosong Yang, Haibo Zhang, Xue Zhao, Wenqing Yao, and Boxing Chen. 2022. Gcpg: A general framework for controllable paraphrase generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4035–4047.

Kayo Yin, Patrick Fernandes, André FT Martins, and Graham Neubig. 2021. When does translation require context? a data-driven, multilingual exploration. *arXiv preprint arXiv:2109.07446*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Appendix

A Related work for discourse paraphrasing

A.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 7) 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 (Section 6).

A.2 Other related work

In this section we discuss a few additional less related papers which were not included in our survey in Section A.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.

B Human evaluation details

In this appendix section, we provide further information of our human evaluation study.

The evaluation is conducted on the platform Label Studio (Tkachenko et al., 2020-2022).[19] Figure 7 shows the interface of our annotation platform. 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. For the evaluation task, we randomly select 20 paraphrases for each of the three lexical codes. 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 8 provides two representative examples for each lexical code that is evaluated in our human study. Qualitative analysis of the annotator comments reveals strengths and shortcomings of DIPPER.

Strengths First, looking at the fifth example in Table 8 with a high lexical diversity code 60, we see that DIPPER is able to preserve the semantic

[19] https://labelstud.io/

meaning of the original text while making significant changes. Second, row 3 shows us that DIPPER is able to leverage the information from the context to increase the diversity of the paraphrase and keep it coherent. 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.

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.

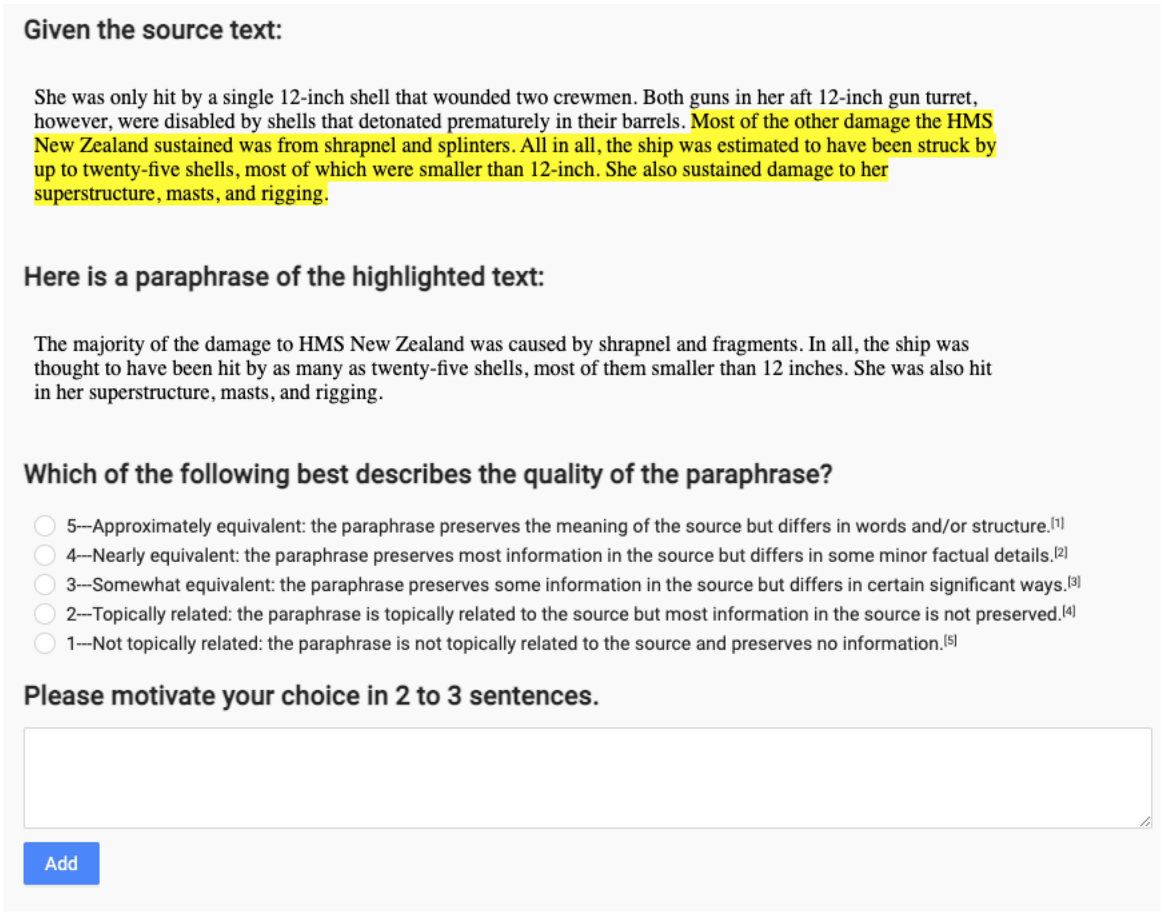
Overall, the human study shows that DIPPER performs well at preserving the semantic meaning of original texts while introducing both semantic and syntactic diversity. Because DIPPER provides user-friendly controllabilty of output diversity, a user can adjust the control code to find the most suitable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

C ROC curves at different FPR

See Figure 8.

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: 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.



Given the source text:

She was only hit by a single 12-inch shell that wounded two crewmen. Both guns in her aft 12-inch gun turret, however, were disabled by shells that detonated prematurely in their barrels. Most of the other damage the HMS New Zealand sustained was from shrapnel and splinters. All in all, the ship was estimated to have been struck by up to twenty-five shells, most of which were smaller than 12-inch. She also sustained damage to her superstructure, masts, and rigging.

Here is a paraphrase of the highlighted text:

The majority of the damage to HMS New Zealand was caused by shrapnel and fragments. In all, the ship was thought to have been hit by as many as twenty-five shells, most of them smaller than 12 inches. She was also hit in her superstructure, masts, and rigging.

Which of the following best describes the quality of the paraphrase?

- 5—Approximately equivalent: the paraphrase preserves the meaning of the source but differs in words and/or structure.[1]
- 4—Nearly equivalent: the paraphrase preserves most information in the source but differs in some minor factual details.[2]
- 3—Somewhat equivalent: the paraphrase preserves some information in the source but differs in certain significant ways.[3]
- 2—Topically related: the paraphrase is topically related to the source but most information in the source is not preserved.[4]
- 1—Not topically related: the paraphrase is not topically related to the source and preserves no information.[5]

Please motivate your choice in 2 to 3 sentences.

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

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 8: Representative model outputs of each lexical code with comments from the annotators (native English teachers/editors). 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.

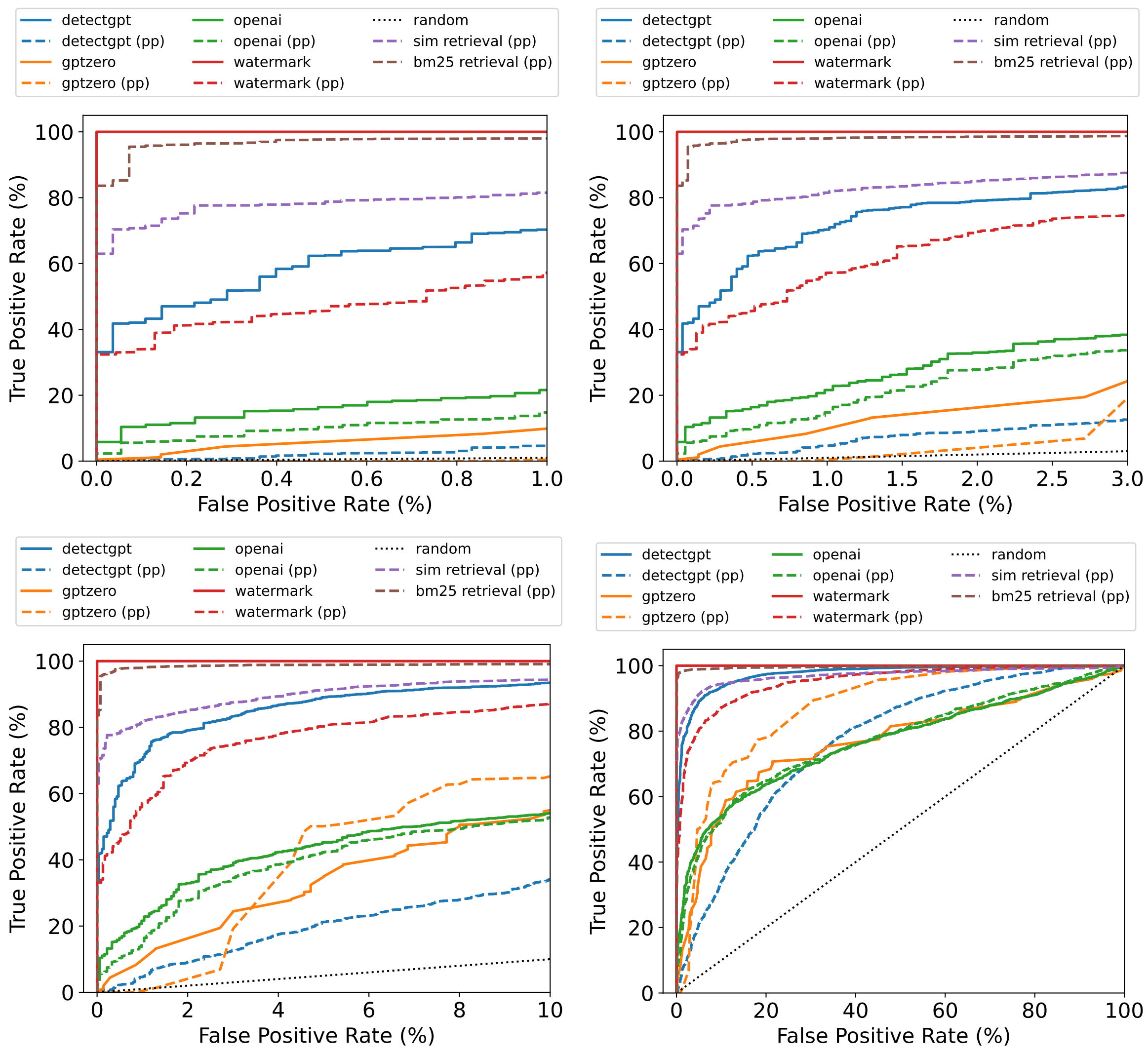


Figure 8: ROC curves for text generated by GPT2-XL, before paraphrasing (solid lines) and after paraphrasing (dashed lines, pp). Different plots represent different clipping thresholds on the X-axis.