Assessing the Efficacy of AI Detectors and Countermeasures in Generative AI

Jason Makai Pindell[1,2], Amarnath Patel[3], Alexander Castronovo[1,2],
Jossaya Camille[3], Zachary Lopez[3], Thandi Menelas[3]

[1]FAU High School in Partnership with Max Planck Academy for Neuroscience
[2]H. L. Wilkes Honors College [3]Florida Atlantic University High School

Wilkes Honors College Symposium, April 4, 2025

1 Abstract

Large language models (LLMs) are increasingly capable of producing text that is difficult to distinguish from human writing, raising concerns about plagiarism and other forms of misuse. This has motivated the development of AI text detectors, which in turn has motivated a growing set of evasion techniques designed to defeat them. In this paper we evaluate two such techniques. The first, iterative sampling, repeatedly paraphrases a passage and checks the result against a detector until a target detectability score is reached. The second, Coherence and Perplexity Based Optimization (CPBO), builds a branching tree of candidate continuations and selects among them by weighting toward higher perplexity while pruning for coherence. We test both approaches against three detectors (RoBERTa, ZeroGPT, and GPTZero) and evaluate the resulting text using perplexity, coherence, and text style transfer (TST) scores. Iterative sampling reduces detectability on older detector architectures, most notably RoBERTa, but its gains largely plateau after the first refinement pass, and repeated paraphrasing steadily erodes stylistic similarity to the original text. CPBO results are preliminary but indicate that the branching procedure reliably surfaces higher perplexity continuations, which detectors depend on as a discriminating signal.

2 1. Introduction

Two risks are commonly associated with the proliferation of LLM generated text. The first is plagiarism, in which generated text is passed off as original human work. The second is model collapse, a phenomenon in which models trained repeatedly on their own generated output experience a narrowing of their output distribution: probable events become over-represented and improbable events are increasingly under-sampled, so that the tails of the true data distribution shrink over successive generations (Shumailov et al., 2024).

AI text detectors were developed to counter both risks by flagging text that is likely to be machine generated. In practice, however, detectors face a range of evasion techniques that can substantially reduce their effectiveness, which motivates the present study.

3 2. Background

3.1 2.1 Detection concepts

We use the following terms throughout this paper. A **token** is a discrete piece of text that a language model processes. **Perplexity** is a measure of how probable a given word is given the preceding context; most perplexity models use a strided sliding window that advances by more than one token at a time (HuggingFace, 2024). **Coherence** refers to the logical flow and readability of a passage. **Text Style Transfer (TST)** quantifies the similarity in style, such as formality or tone, between two texts.

3.2 2.2 Existing evasion techniques

Several evasion strategies have been proposed in prior work. Single shot paraphrasing rewrites a passage once with an LLM; Krishna et al. show that this can successfully evade detection, although retrieval based defenses can partially counter it (Krishna et al., 2023). SICO (Substitution based In Context example Optimization) constructs prompts designed to evade detection directly, but it depends on a proxy detector during optimization and produces comparatively poor TST scores (Lu et al., 2023).

We independently proposed an iterative sampling approach in March 2024. As of January 2025, we found that a closely related iterative procedure had been described by Sadasivan et al., published in a revised 2025 version

of their manuscript on the reliability of AI text detection (Sadasivan et al., 2025).

4 3. Objective

Our goal is to maximize the perplexity, coherence, and TST of AI generated output text in order to stress test AI detection algorithms and characterize where their discriminating signal breaks down.

5 4. Methods

5.1 4.1 Iterative sampling

The iterative sampling procedure operates in four steps, illustrated in Figure 1.

1. **Initial paraphrasing.** A paraphrase prompt is applied to the input text, and an initial paraphrased version is generated by the model.
2. **Iterative refinement.** The AI generated output is fed back into the model and the paraphrase prompt is reapplied.
3. **Detectability check.** The paraphrased text is evaluated for detectability at each pass; this continues for ten iterations.
4. **Final output.** Once a target detectability criterion is met, the text is finalized. The result is a rewritten passage with reduced traceability to the original.

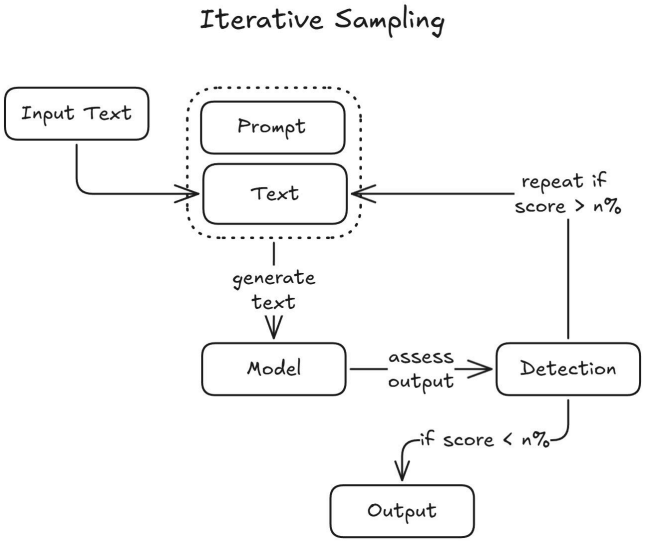

Figure 1: Overview of the iterative sampling procedure. A paraphrase prompt is repeatedly reapplied to the model's own output and checked against a detector until a target detectability score is reached.

5.2 4.2 Coherence and Perplexity Based Optimization (CPBO)

CPBO takes a different approach, building a decision tree of candidate continuations rather than repeatedly rewriting a full passage.

In the initial prediction stage, the model predicts the top P most likely next steps, and for each of those steps it generates the top P subsequent steps in turn. This continues for an arbitrary number of strides, with each stride consisting of two tokens (Figure 2). A secondary model then grades the coherence of each branch, and branches are trimmed according to that model's predictions.

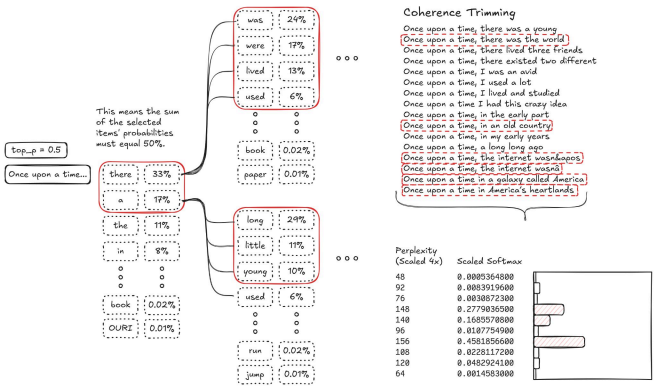

Figure 2: CPBO decision tree at a single time step. Branches are generated from each token's top P candidates, and the tree branches again at each subsequent step.

Branch selection is performed by applying a softmax function to the perplexity scores of the candidate branches and then randomly selecting a root node according to that softmax weighting, which biases selection toward higher perplexity continuations. The selected branch is then extended one step at a time, root selection is repeated using the updated weighted softmax, and this process continues until the model completes generation.

We note, in retrospect, that this branching and pruning structure is conceptually related to beam search. The key difference is one of objective: conventional beam search retains the highest probability, that is lowest perplexity, branches at each step in order to maximize the likelihood of the generated sequence, whereas CPBO deliberately biases toward higher perplexity branches, since our goal is to raise perplexity high enough to defeat perplexity sensitive detectors, with a secondary coherence check used to keep the resulting text readable.

5.3 4.3 Secondary coherence model

The secondary model used for branch trimming was trained using supervised RLHF on a text corruption task. Coherent text was extracted from Wikipedia and Project Gutenberg, split into segments, and corrupted using several methods, including syntax breaking, nonsense insertion, structural reordering, semantic drift, and tense shifting, following the poor text construction approach of Sharma et al. (2024). We compared three candidate architectures, BERT, RoBERTa, and GPT2, for the coherence scoring task (Figure 3).

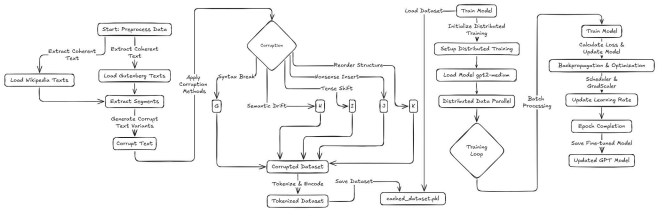

Figure 3: Architecture for the secondary predictive coherence model, trained with supervised RLHF on a corrupted text dataset.

5.4 4.4 Experimental setup

Human generated text corpora were obtained from Kaggle. Three models, Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o, were used to paraphrase the human generated text with no special prompt, producing 35 AI augmented passages. Each passage was then humanized using both iterative sampling and CPBO. Outputs at each iteration were analyzed across three detection methods, RoBERTa 105M, ZeroGPT, and GPTZero. Iterations were compared using word frequency, perplexity, coherence following the entity based approach of Barzilay and Lapata (2005), and TST. TST was evaluated using BLEU, which compares overlapping n-grams between two texts, following the general framing of style transfer evaluation used by Mir et al. (2019).

6 5. Results

6.1 5.1 Detection scores under iterative sampling

Across all three paraphrasing models, detection scores from RoBERTa dropped sharply after the first paraphrase and continued a shallow downward trend over the following ten iterations (Figure 4; R-squared values of 0.022, 0.024, and 0.013 for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o respectively).

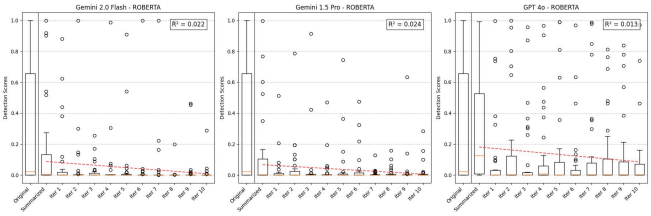

Figure 4: Detection scores across original, single shot paraphrased, and ten iterative sampling passes, evaluated with RoBERTa, for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

ZeroGPT scores also declined with iteration, though with more variance and a weaker overall fit (Figure 5; R-squared of 0.070, 0.018, and 0.012 respectively).

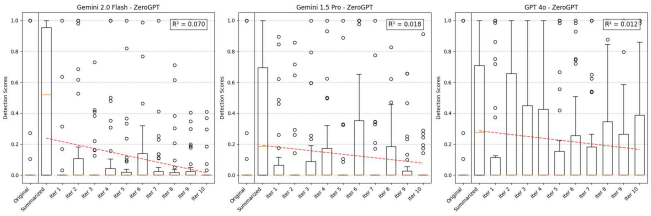

Figure 5: Detection scores across original, single shot paraphrased, and ten iterative sampling passes, evaluated with ZeroGPT, for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

GPTZero, by contrast, remained largely saturated near a detection score of 1.0 across all iterations and all three paraphrasing models, with very low R-squared values (0.004, 0.004, and 0.008), indicating that iterative sampling had essentially no effect on this detector (Figure 6).

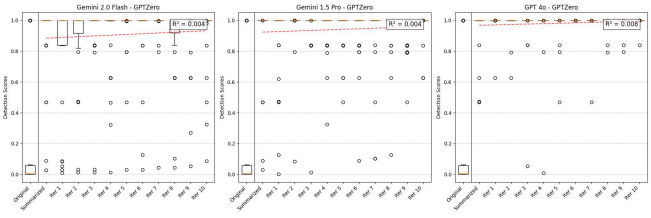

Figure 6: Detection scores across original, single shot paraphrased, and ten iterative sampling passes, evaluated with GPTZero, for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

6.2 5.2 Perplexity

Perplexity scores across the ten iterations remained roughly flat for all three models, with weak positive trends and low R-squared values (0.006, 0.004, and 0.013 for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o; Figure 7).

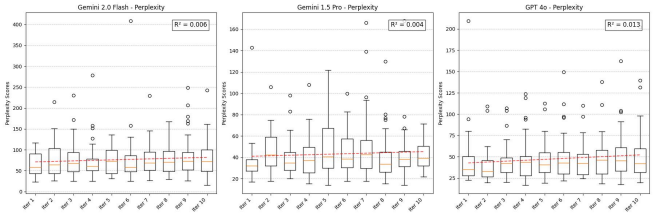

Figure 7: Perplexity scores across ten iterations of iterative sampling for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

6.3 5.3 Coherence

Coherence, scored with the Barzilay and Lapata method, showed a mild decreasing trend for Gemini 2.0 Flash (R-squared of 0.083) but a mild increasing trend for the larger models, Gemini 1.5 Pro and GPT 4o (R-squared of 0.008 and 0.020 respectively), suggesting that higher parameter models tend to preserve or even improve coherence over repeated paraphrasing while the smaller model does not (Figure 8).

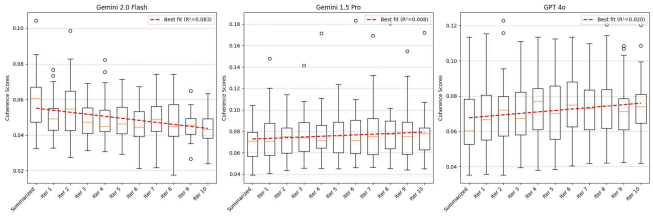

Figure 8: Coherence scores, evaluated with the Barzilay and Lapata (2005) method, across ten iterations of iterative sampling for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

6.4 5.4 Text style transfer

BLEU based TST scores declined across iterations for all three models (R-squared of 0.110, 0.048, and 0.044 for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o respectively), indicating that repeated paraphrasing steadily moves the output further from the stylistic profile of the original passage (Figure 9).

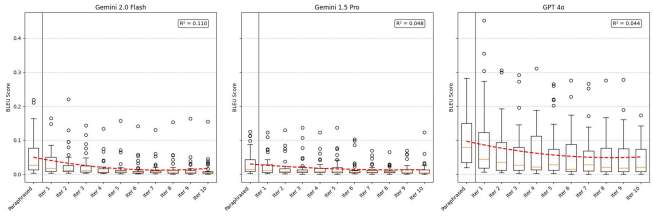

Figure 9: BLEU based TST scores across ten iterations of iterative sampling for Gemini 2.0 Flash, Gemini 1.5 Pro, and GPT 4o.

6.5 5.5 CPBO

For the coherence trimming task, the three candidate secondary model architectures achieved F1 scores of 0.73 for BERT, 0.75 for RoBERTa, and 0.65 for GPT2, indicating that RoBERTa was the strongest of the three at distinguishing coherent from corrupted text.

Initial observations of the branching process show clear bursts of higher perplexity at particular steps within a branch (Figure 10), and this pattern persists after branches are trimmed using the RoBERTa based coherence trimming algorithm (Figure 11). Because of the computational cost of running the full branching procedure, this dataset is still being collected and these results should be considered preliminary.

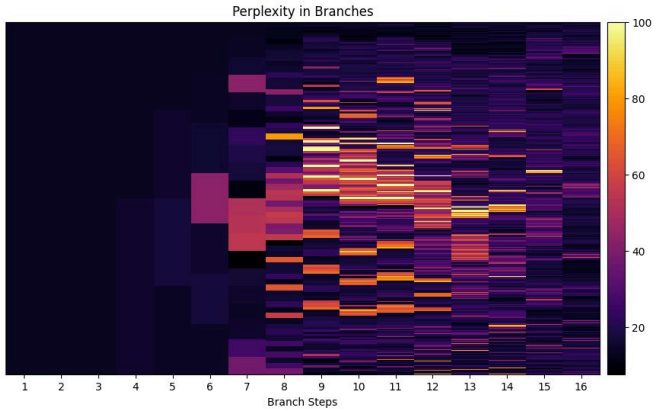

Figure 10: CPBO perplexity at each step of the branch, before trimming. Brighter regions indicate higher perplexity. Top K = 5, stride size of two tokens, eight steps total.

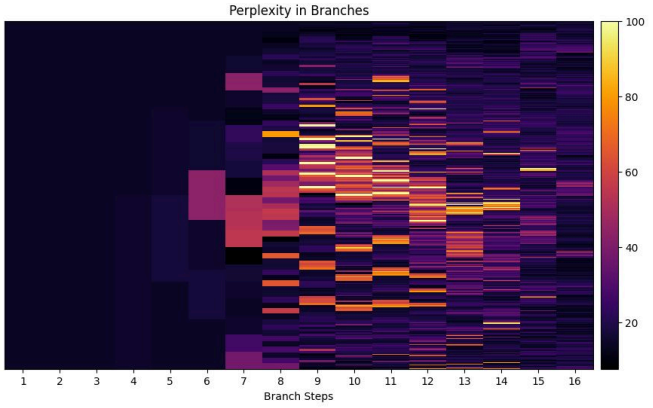

Figure 11: CPBO perplexity at each step of the branch, after trimming with the RoBERTa coherence trimming algorithm. Same parameters as Figure 10.

7 6. Discussion

7.1 6.1 Findings and contributions

Iterative sampling reduces detectability most clearly on older detector architectures such as RoBERTa. Its effect on coherence differs by model scale: higher parameter models (GPT 4o, Gemini 1.5 Pro) tend to show a slight increasing coherence trend across iterations, while the lower parameter model (Gemini 2.0 Flash) shows a decreasing trend. Across all three models, TST declines steadily over successive iterations, and detectability shows no substantial further improvement after the first refinement pass, meaning most of the benefit of iterative sampling is realized immediately rather than accruing gradually.

For CPBO, the branch selection procedure reliably favors regions of higher perplexity, which is consistent with its design goal, but the dataset remains preliminary and needs a fuller evaluation before strong conclusions can be drawn about its effect on detectability.

7.2 6.2 Limitations and further study

These results are limited by incomplete computational resources, and several conclusions, particularly those concerning CPBO, remain tentative until further data is collected. Our findings are also specific to the three detectors we evaluated (RoBERTa 105M base, ZeroGPT, and GPTZero) and may not generalize to other detection architectures, including watermarking based approaches, which embed a statistical signal directly into generated text rather than inferring one after the fact (Dathathri et al., 2024).

Future work includes counter training a detection model directly on outputs produced by these evasion methods, applying mechanistic interpretability techniques to better understand why particular detectors succeed or fail, and testing both evasion methods across a wider range of text styles and languages. Related work on how human edited paraphrases interact with detector performance may also inform this direction (Lau & Zubiaga, 2024).

8 Acknowledgements

We thank the FAU Office of Undergraduate Research Inquiry (OURI) for their generous grant, and the FAU Office of Information Technology (OIT) for their rapid support.

9 References

Barzilay, R., & Lapata, M. (2005). Modeling local coherence: An entity based approach. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05) (pp. 141-148). Association for Computational Linguistics. https://aclanthology.org/P05-1018/

Dathathri, S., See, A., Ghaisas, S., Huang, P., McAdam, R., Welbl, J., Bachani, V., Kaskasoli, A., Stanforth, R., Matejovicova, T., Hayes, J., Vyas, N., Merey, M. A., Bunel, R., Balle, B., Cemgil, T., Ahmed, Z., Stacpoole, K., Shumailov, I., … Kohli, P. (2024). Scalable watermarking for identifying large language model outputs. Nature, 634(8035), 818-823. https://doi.org/10.1038/s41586-024-08025-4

HuggingFace. (2024). Perplexity of fixed length models. https://huggingface.co/docs/transformers/en/perplexity

Krishna, K., Song, Y., Karpinska, M., Wieting, J., & Iyyer, M. (2023). Paraphrasing evades detectors of AI generated text, but retrieval is an effective defense. arXiv. https://arxiv.org/abs/2303.13408

Lau, H. T., & Zubiaga, A. (2024). Understanding the effects of human written paraphrases in LLM generated text detection. arXiv. https://arxiv.org/abs/2411.03806

Lu, N., Liu, S., He, R., Wang, Q., Ong, Y., & Tang, K. (2023). Large language models can be guided to evade AI generated text detection. arXiv. https://arxiv.org/abs/2305.10847

Mir, R., Felbo, B., Obradovich, N., & Rahwan, I. (2019). Evaluating style transfer for text. arXiv. https://arxiv.org/abs/1904.02295

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., & Feizi, S. (2025). Can AI generated text be reliably detected? arXiv. https://arxiv.org/abs/2303.11156

Sharma, V., Padthe, K., Ardalani, N., Tirumala, K., Howes, R., Xu, H., Huang, P., Li, S., Aghajanyan, A., Ghosh, G., & Zettlemoyer, L. (2024). Text quality based pruning for efficient training of language models. arXiv. https://arxiv.org/abs/2405.01582

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., & Gal, Y. (2024). AI models collapse when trained on recursively generated data. Nature, 631(8022), 755-759. https://doi.org/10.1038/s41586-024-07566-y