Beyond DetectGPT: Towards AI-generated Text Detection in a Black-box Setting and with Lighter Computation

Bale Chen, Corey Chen, Richen Du, Manli Zhao
New York University
{bale.chen, coreychen, rd2526, mz3548}@nyu.edu

Abstract

Recently, we have witnessed many releases of highly capable large natural language generation (NLG) models, which elicits potential issues like plagiarism. It is essential to develop a reliable detector to distinguish between human and machine-generated text. Our paper builds upon the DetectGPT framework (Mitchell et al., 2023) and generalizes it to a more challenging black-box setting where we cannot access the model that generates the content[1]. We proposed a fine-tuned OPT approach that consistently outperforms other methods in the black-box setting but fails to match up with the white-box DetectGPT. We also analyzed potential reasons for the poor performance of the ensemble method and noising perturbation. Lastly, accuracy under false positive control (AFC) is shown to be a better metric than AUROC score in the context of AI-generated text detection. Our findings show that the fine-tuned OPT method is an efficient approach to the black-box AI-generated text detection task and has the potential to generalize DetectGPT to a black-box setting.

1 Introduction

The use of NLG models has seen a meteoric rise over the past few months and shown remarkable results in performing a wide array of tasks (Bhaskar et al., 2022; Hendy et al., 2023; Lamichhane, 2023). However, with this capability also comes caution. These applications have the ability to produce potentially false or malicious social media content, academic dishonesty, and authorship violations. It is important to discriminate between AI-generated text and human-written text so that we can avoid these serious pitfalls. Automated AI-generated text detection, especially detection methods that don't require extensive supervised samples and training, is thus studied by multiple researchers (Jawahar et al., 2020; Kirchenbauer et al., 2023; Mitchell et al., 2023; Gehrmann et al., 2019; Chakraborty et al., 2023).

The automated detection task can be set up in two ways: the white-box setting and the black-box setting. In a white-box setting, detectors are allowed to access the source model from which the text is generated. However, in real-world scenarios, we are typically either unaware of or lack access to the source model. Detecting whether a piece of text is AI-generated or not without accessing the source model is what we define as the black-box setting. The black-box setting is a more general yet challenging task and is one of the main focuses of this paper.

Specifically, this project builds upon the white-box setting and zero-shot detection framework DetectGPT proposed by Mitchell et al. (2023). A detailed explanation of DetectGPT mechanism and limitations can be found in Section 3. Our objectives are to generalize DetectGPT to a black-box setting and lower the computation cost. The main contributions of our research are that we:

- Proposed the fine-tuned OPT method that shows potential of generalizing DetectGPT framework to a black-box setting. The failure of ensemble method mentioned in Mitchell et al. (2023) is also explained.
- Analyzed the reasons behind the failure of model-free perturbation that aims to lower the computational complexity of DetectGPT.
- Revealed the severe false positive rate disguised by the AUROC used in DetectGPT and adopted a more practical metric, AFC.

2 Related Work

2.1 Zero-shot Detectors

Zero-shot detectors are a type of detector that does not require training or fine-tuning to perform detection (Jawahar et al., 2020). They are shown to be more adaptive than supervised detec-

[1]Our codebase is open-source on GitHub.

tors across languages and text domains. For example, Solaiman et al. (2019) proposed the total log-probability method that assumes that machine-generated text would have a higher log-probability given the model parameters since when language models generate text, they choose to generate tokens that maximize log-probability. Gehrmann et al. (2019)'s GLTR builds upon this assumption by developing token-level measures to visualize the disparity between GPT-generated text and human-written text. Dou et al. (2021) developed a framework that categorizes and visualizes perceivable errors in both machine and human text samples, showing that machine-generated text has semantic distribution that is distinguishable from human-written text.

2.2 Black-box Detectors

Training or fine-tuning a separate model for an AI-generated text detection task usually falls in the black-box setting where we do not access the source model that generates the text. Fagni et al. (2020) consolidated a dataset of fake tweets generated by 23 different models and 17 human sources. They trained different neural networks to detect AI-generated tweets and found that fine-tuned large language models tend to perform better. Ippolito et al. (2019) used fine-tuned BERT and bag-of-word features to detect machine-generated text under different decoding strategies. Previous literature that fall in the black-box setting mainly involves extensive training or fine-tuning, while Bakhtin et al. (2019) and Uchendu et al. (2020) warn that classifiers trained explicitly for detecting AI-generated text are prone to overfitting the training set distribution, which is empirically validated in Mitchell et al. (2023).

To our knowledge, previous literature lacks exploration into zero-shot detectors in the black-box setting, which is what our research aims to contribute towards.

3 Overview and Limitations of DetectGPT Framework

Mechanism DetectGPT is a zero-shot white-box detection method that uses the source model itself to evaluate if a passage came from its own samples by computing the average drop in log-likelihood (perturbation discrepancy) after applying perturbations. One perturbation is done by first masking out a portion of the text and then filling the masks using T5-3b model. The intuition behind DetectGPT is that if the text is generated by the source model, then it would approximately maximize the log-likelihood given the source model parameters. When we change a portion of the model-generated text following the sampling distribution of a different model, the log-likelihood given the source model parameters would decrease. Human-written text, however, does not necessarily fall on the local maxima of the log-likelihood curve, so perturbing will not always lead to drops in log-likelihood. Thus, the log-likelihood difference between the perturbed text and original text can distinguish machine-generated and human-written text. The perturbation discrepancy is mathematically formulated as

$$d(x, p_\theta, q) = \log p_\theta(x) - \mathbb{E}_{\tilde{x}\sim q(\cdot|x)}[\log p_\theta(\tilde{x})] \quad (1)$$

where $x \sim p_{\theta_0}(\cdot)$ is a piece of text generated by the source model p_{θ_0}, p_θ represents the scoring model with parameter θ, and $q(\cdot \mid x)$ represents the perturbation distribution given x (i.e. the T5-3b mask-filling). Scoring model is the model with which we compute the log-likelihood. Thus, DetectGPT's assumption is that if we use the source model as the scoring model (i.e. $\theta = \theta_0$), then $\mathbb{P}(\log p_{\theta_0}(x) > \log p_{\theta_0}(\tilde{x}))$ is high. With this property, $d(x, p_{\theta_0}, q)$ is shown to be discriminative between machine-generated and human-written text.

Limitations The experiment results showed that detection performance degenerates when the scoring model is not the source model (i.e. $\theta \neq \theta_0$), which the black-box setting requires. Another limitation is its intensive computation compared to other zero-shot detection methods. In practice, to generate a low-variance measure of perturbation discrepancy, perturbation is run 100 times to compute the normalized discrepancy for one piece of text, requiring approximately 20 minutes. The study suggested looking into potential alternatives for a more efficient perturbation function to reduce costs.

4 Methods

For black-box setting, we implemented the ensemble method mentioned in Mitchell et al. (2023) and proposed the fine-tune OPT method in Section 4.1. We also proposed the model-free perturbation function explained in Section 4.2.

4.1 Black-box setting

Ensemble Scoring Following the observation in the DetectGPT experiments, some models produce better detection performance when used as scoring model. Intuitively, using an ensemble method would leverage the diverse capability of different scoring models. We used a basic ensemble method that calculates the average perturbation discrepancy of three scoring models for each piece of text. We did not investigate further on more complex ensemble methods because it is not promising as explained in Section 6.

Fine-tuned OPT Since the DetectGPT rationale is that the machine-generated text maximizes the log-probability given the model parameter, the key issue about the black-box setting is model parameters are inaccessible. Thus, we propose using the Open Pretrained Transformer (OPT) model (Zhang et al., 2022) that is fine-tuned on machine-generated text from the source model as the scoring model. The hypothesis is that the fine-tuned OPT is updated to the proximity of the source model in the parameter space. Thus, the fine-tuned OPT acts as a proxy model of the inaccessible source model. We chose the pretrained OPT-1.3B from the HuggingFace models[2]. The detailed fine-tuning configuration can be found in Appendix A.1.

4.2 Model-free Perturbation function

We propose a noising method that adds noise to the embedding layer or the final hidden layer when scoring the log-likelihood, which only takes seconds to run and is model-free. The hypothesis behind this is that the random noise would induce an equivalent log-likelihood drop to the T5 mask filling method. The perturbation discrepancy with noising can be formulated as:

$$d_{\text{noise}}(x, p_\theta, q) = \log p_\theta(x) - \mathbb{E}_{\epsilon\sim q}[\log p_\theta(x+\epsilon)] \quad (2)$$

where q is either Gaussian or Uniform distribution. Implementation details can be found in Appendix A.2.

5 Experiments

In this section, we present the results of main experiments. Specifically, we first study whether the ensemble and fine-tuned OPT method would generalize the DetectGPT method to a black-box setting in Section 4.1. Then, in Section 4.2, we probe into whether noise perturbation can match with the T5 mask filling performance. We tested their performance across different text domains and source models. GPT-2 (Radford et al., 2019), GPT-Neo (Black et al., 2021), GPT-J (Wang, 2021), and Neo-X (Black et al., 2022) are used as the source models of increasing sizes. Total log probability (Solaiman et al., 2019) and the original DetectGPT method are used as baselines since we aim to match the white-box detectors using our methods in a black-box setting.

Data We used 3 datasets to cover various domains. The XSum dataset (Narayan et al., 2018) contains news articles. The SQuAD dataset (Rajpurkar et al., 2016) includes a collection of question-answer pairs from Wikipedia articles. The Reddit WritingPrompts dataset (Fan et al., 2018) is comprised of human-written stories with writing prompts. Note that these datasets are all human-written text. We prompt the source models with the first 30 tokens of each piece of text to generate a machine-generated counterpart. This setup is identical to the original DetectGPT paper.

Evaluation Metric In real-world scenarios (e.g. detecting plagiarism), it is undesirable to accuse an innocent person as opposed to exonerating a guilty person. In other words, we expect a good detector to almost never classify human-written text as machine-generated, which means the false positive rate (FPR) of the detection performance should be minimal. Thus, we used a similar approach to Krishna et al. (2023) that calculates accuracy after fixing the FPR to 1%. We adjusted the classification threshold (the threshold of perturbation discrepancy that any text with perturbation discrepancy higher than that is classified as machine-generated) to fix the FPR. This metric is what we define as accuracy under false positive control (AFC). Note that if the perturbation discrepancy of machine-generated text and human-written text are distributed identically, the lower bound of AFC is roughly 0.50.

5.1 Main Results

DetectGPT in the Black-box Setting Figure 1 shows the AFC of DetectGPT in a black-box setting across source-score model pairs. On the diagonal line, the source and scoring model are the same, which belongs to the white-box setting as the original DetectGPT method, while the off-diagonal entries are under a black-box setting. We can see that

[2]https://huggingface.co/facebook/opt-1.3b

Method	XSum					SQuAD					WritingPrompts				
	GPT-2	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	Neo-2.7	GPT-J	NeoX	**Avg.**	GPT-2	Neo-2.7	GPT-J	NeoX	**Avg.**
$\log p(x)$	0.53	0.56	0.53	0.51	0.54	0.65	0.56	0.53	0.51	0.56	0.72	0.61	0.65	0.58	0.64
DetectGPT-WB	**0.87**	**0.91**	**0.65**	**0.62**	**0.76**	**0.87**	**0.77**	0.54	0.51	**0.67**	**0.91**	**0.84**	**0.69**	**0.73**	**0.79**
DetectGPT-BB	0.60*	0.55	0.60*	0.55*	0.58	0.52	0.52	0.53	0.52	0.52	0.62	0.58	0.61	0.58	0.60
Ensemble	0.50	0.50	0.51	0.51	0.50	0.48	0.49	0.50	0.51	0.49	0.50	0.51	0.53	0.52	0.51
Fine-tuned OPT	0.59	0.62*	0.56	0.54	0.58	0.66*	0.65*	**0.55***	**0.55***	0.60*	0.63*	0.64*	0.64*	0.60*	0.63*

Table 1: AFC for detecting samples from the source model on the given dataset for our ensemble-DetectGPT, fine-tuned OPT and baseline (500 samples used for evaluation). **Bold** shows the best AFC within each column (model-dataset combination); asterisk (*) denotes the best AFC in the black-box setting (i.e. methods under the middle line). WB stands for white-box, and BB stands for black-box.

for each source model, the performance is rather poor when the score model is different from the source model, since the AFC is very close to 0.50. We also observe that the detector's performance slightly degrades when the source model is larger. The larger score models (GPT-J and Neo-X) cannot identify AI-generated text at all in the black-box setting and performs relatively poorly even in the white-box setting.

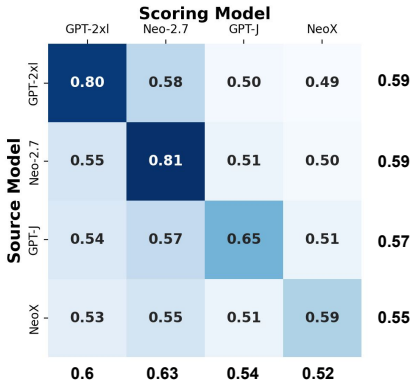


Figure 1: AFC of DetectGPT for using a different model to score the texts than the source model that generated the passage. The AFC is averaged for the three datasets.

Ensemble Method and Fine-tuned OPT We present the comparison of different detection methods in Table 1. The source models are presented in an increasing sizes from left to right under each dataset. The total log-probability method log p(x) and DetectGPT-WB are the two white-box baseline methods. DetectGPT-BB means performing the original DetectGPT method but using a scoring model that is different from the source model. The AFC reported in the DetectGPT-BB row is the best AFC achieved by one of the three scoring models.

The ensemble method completely fails with AFC around 0.50. The averaged perturbation discrepancy is thus nondiscriminatory between AI-generated text and human-written text. Our fine-tuned OPT method performs as well as the Detect-

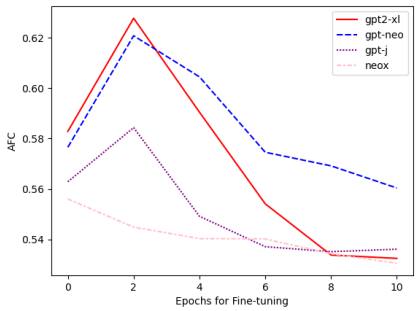
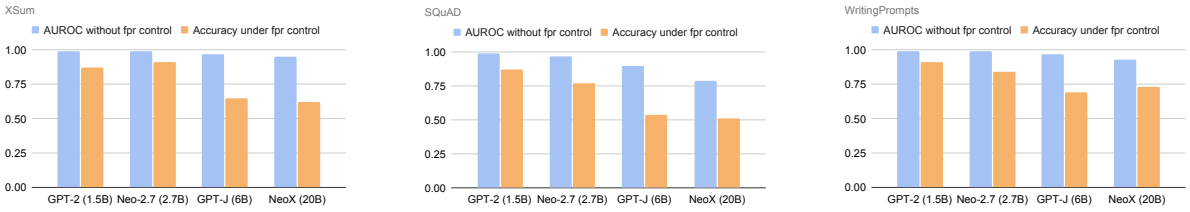


Figure 2: AFC of OPT fine-tuned for different epochs. The four lines represent the four source models and the AFC is averaged for all three datasets.

GPT method in the black-box setting on the XSum dataset, and it consistently outperforms DetectGPT-BB on the other two datasets. This result indicates that fine-tuned OPT is the best detection method in the black-box setting. Although fine-tuned OPT also outperforms DetectGPT in the white-box setting on SQuAD dataset with GPT-J and NeoX as source models, overall it cannot match up with the white-box detector's performance.

Effect of Fine-tuning Since the fine-tuned OPT approach shows the best performance in black-box setting, we examine whether the fine-tuning process actually contributes to the performance improvement. From Figure 2, fine-tuning for 2 epochs would increase the AFC compared with OPT without fine-tuning. This result indicates that fine-tuning scoring model on the text generated by the source model does improve the DetectGPT framework when the scoring model is different from the source model. We also observe that fine-tuning for more than 2 epochs lead to downgrading performance possibly due to overfitting. Also, fine-tuning OPT on text generated by larger models GPT-J and Neo-X shows lower or no AFC improvement.

Model-free Perturbation The results of model-free perturbation functions are presented in Table 2. We observed that perturbing text by adding



(a) XSum (b) SQuAD (c) WritingPrompts

Figure 3: Comparision between AUROC and AFC metrics on different model sizes and datasets.

	Source Model				
Method	GPT-2	Neo-2.7	GPT-J	NeoX	**Avg.**
Mask filling	**0.88**	**0.84**	**0.63**	**0.62**	**0.74**
Perturb-G	0.51	0.51	0.52	0.51	0.51
Perturb-U	0.50	0.52	0.49	0.50	0.50

Table 2: Average AFC across three datasets for each source model. Mask filling is the original perturbation function used in DetectGPT. Perturb-G uses Gaussian noise and Perturb-U uses Uniform noise.

Gaussian or Uniform noise resulted in significantly poorer performance compared to using the T5-3b model to fill masks. Adding noise does not help induce the differing perturbation discrepancy. Moreover, there was minimal distinction between Gaussian and Uniform noise.

6 Discussion

Black-box Detection According to Figure 1, the performance of using a different scoring model from the source model, namely in the black-box setting, is very poor where AFC is near 0.5. The perturbation discrepancy in the black-box setting is thus not meaningful for detecting machine-generated text. This is likely the reason why the ensemble method fails, since in this case, the ensemble method is aggregating useless information.

On the contrary, the fine-tuned OPT method performs consistently better than DetectGPT method in the black-box setting. As the experiment shows, fine-tuning a different scoring model OPT for just two epochs on the text generated by various source models can improve detection performance. This result provides empirical evidence for the hypothesis – fine-tuning updates the OPT's parameter to be similar with the source model parameters. While this is difficult to rigorously validate, fine-tuning scoring model shows potential of generalizing DetectGPT's framework to the black-box setting. Regarding the source model size, fine-tuning results in less improvement on larger source models. This is possibly caused by the fact that we used OPT-1.3B as the scoring model to fine-tune, which is small compared to GPT-J and Neo-X. To estimate the parameters of larger models, the scoring model may also need to be larger.

Failure of Model-free Perturbation A possible reason for the failure of noising is that natural language tokens are discrete. To create a small shift on the log-likelihood curve, changing the tokens with tokens of similar meaning, such as mask filling perturbation, works better. Adding random noise to either the embedding vectors or the final hidden state might result in a place on the log-likelihood curve that is poorly defined since it does not represent any meaningful token, which produces uncertain change of the log-likelihood that violates the rationale of DetectGPT method.

False Postive Rate Control An interesting finding from using AFC as the evaluation metric is that the performance of DetectGPT reduces from Mitchell et al. (2023). Figure 3 shows that if we control FPR to be 1%, the accuracy of DetectGPT drops significantly when the source model increases in size, especially for models on the SQuAD dataset. This suggests that to evaluate the performance of an AI-generated text detector, AUROC might not be reliable. High AUROC performance may come at the price of high FPR, leading to human-written content being falsely labelled.

7 Conclusion

In conclusion, in the black-box setting, the DetectGPT method performs rather poorly, which causes the ensemble scoring model to fail as well. Our paper shows that fine-tuning OPT as the scoring model on text generated by the source model achieves the best performance across detectors in the black-box setting, which shows potential in generalizing DetectGPT to the black-box setting. Moreover, adding random noise in the scoring model is not performance-wise equivalent to the original mask-filling perturbation method, which leaves the high computation cost unsolved.

References

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

Adithya Bhaskar, Alexander R. Fabbri, and Greg Durrett. 2022. Zero-shot opinion summarization with gpt-3. *ArXiv*, abs/2211.15914.

Sid Black, Stella Rose Biderman, Eric Hallahan, Quentin G. Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Martin Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Benqi Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model. *ArXiv*, abs/2204.06745.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Rose Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow.

Souradip Chakraborty, A. S. Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. *ArXiv*, abs/2304.04736.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2021. Is gpt-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text. In *Annual Meeting of the Association for Computational Linguistics*.

Tiziano Fagni, F. Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. 2020. Tweepfake: About detecting deepfake tweets. *PLoS ONE*, 16.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. Gltr: Statistical detection and visualization of generated text. In *Annual Meeting of the Association for Computational Linguistics*.

Amr Hendy, Mohamed Gomaa Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *ArXiv*, abs/2302.09210.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2019. Automatic detection of generated text is easiest when humans are fooled. In *Annual Meeting of the Association for Computational Linguistics*.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense.

Bishal Lamichhane. 2023. Evaluation of chatgpt for nlp-based mental health applications. *ArXiv*, abs/2303.15727.

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

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *ArXiv*, abs/1908.09203.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Conference on Empirical Methods in Natural Language Processing*.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. `https://github.com/kingoflolz/mesh-transformer-jax`.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Collaboration Statement

Bale designed the experiments, wrote the code and subsections in paper for model-free perturbation function and fine-tuned OPT, and ran the experiments with Manli. Manli also wrote the code and subsections in the paper for ensemble method. Corey contributed to the literature review, and drafted the paper. Richen wrote the code and subsections in the paper about controlling the false positive rate. All members contributed to the paper proofreading and editing.

A Appendix

A.1 Fine-tuning Configuration

For each source model, we prompt it with first 30 tokens of each piece of text in the three datasets. This results in 3172 pieces of text. Each piece of text has on average 312 tokens. We then concatenate all the text and split it into trunks that has the uniform length of 128 tokens. This is because we want to unify the length each data point in the training batch, but we don't want to add extraneous padding tokens or truncate the useful information.

Then, we fine-tune a pretrained OPT-1.3b model on this consolidated dataset. We use the Adam optimizer with initial learning rate at 2×10^{-5}. The batch size is set at 32. Eventually, for each of the four source models in our experiment, we have an OPT model that is fine-tuned on the text generated by the source model.

A.2 Random Noising

Note that when scoring the log-likelihood, we pass a piece of text into the scoring model. We thus experimented with two places of adding random noise to the scoring model. First, we add the noise tensor to the input embeddings. Namely, after the tokens are transformed into embedding tensors, we randomly choose 30% of the embedding tensors and add the noise to it. Another method is to add the random tensor to the final hidden state of the scoring model. The perturbation discrepancy is thus the difference between the log-likelihood scored with and without the random noise. For the random tensor ϵ, we keep $\mathbb{E}[\epsilon] = 0$ and $\text{Var}(\epsilon) = 1$, because Mitchell et al. (2023) frames the perturbation function as adding a random variable that has mean zero and variance 1 to the original input text.