Smaller Language Models are Better Zero-shot Machine-Generated Text Detectors

Anonymous EMNLP submission

Abstract

As large language models are becoming more embedded in different user-facing services, it is important to be able to distinguish between human-written and machine-generated text to verify the authenticity of news articles, product reviews, etc. Thus, in this paper we set out to explore whether it is possible to use one language model to identify machine-generated text produced by another language model, in a zero-shot way, even if the two have different architectures and are trained on different data. We find that overall, *smaller models are better universal machine-generated text detectors*: they can more precisely detect text generated from both small and larger models, without the need for any additional training/data. Interestingly, we find that whether or not the detector and generator models were trained on the same data is not critically important to the detection success. For instance the OPT-125M model has an AUC of 0.90 in detecting GPT4 generations, whereas a larger model from the GPT family, GPTJ-6B, has AUC of 0.65.

1 Introduction

With the rapid improvement in fluency of the text generated by large language models (LLMs), these system are being adopted more and more broadly in a wide range of applications, including chatbots, writing assistants, and summarizers. Generations from these models are shown to have human-like fluency (Liang et al., 2022; Yuan et al., 2022), making it difficult for human readers to differentiate machine-generated text from human-written text. This can have significant ramifications, as such LLM-based tools can be abused for unethical purposes like phishing, astroturfing, and generating fake news (He et al., 2023). As such, we need to be able to reliably and automatically detect machine generated text.

While there has been work on training specialized classifiers for distinguishing between machine-generated text of specific models and human-written text (Verma et al., 2023; OpenAI), such approaches are not always applicable as access to training data might be limited, the classifier might overfit to a given model's generation, and it would need to be constantly updated to account for distribution shifts. As such, zero-shot methods are developed that can detect machine generated text without any training, using the generator model and its likelihood distribution over tokens (Mitchell et al., 2023; Gehrmann et al., 2019; Solaiman et al., 2019; Ippolito et al., 2020). In practice, however, we often need to detect machine-generations in situations where we do not know which model could have been used as the text generator — and even if we do know the generator, we might not have white-box access to it or its logits, or access might be behind a paywall (e.g. GPT3).

Therefore, in this paper we set out to explore the zero-shot detection of machine-generated text without any knowledge of the generator, or access to it. We do this by exploring whether it is possible to use signals from one language model (a *detector* model) to identify machine-generated text generated by another language model (*the generator*). We use surrogate detector models, whose likelihood functions we do have access to, and run zero-shot tests using different signals such as likelihood, rank, log rank, and curvature of the detector model over text (Ippolito et al., 2020; Gehrmann et al., 2019; Mitchell et al., 2023) to distinguish between machine-generated and human written text. We call this cross-detection.

We conduct an extensive empirical analysis by experimenting on a slew of models with different sizes (from tens of millions to billions of parameters), architectures (GPTs, OPTs, Pythias) and pre-training data (Webtext and the Pile). Our main finding is that zero-shot *cross-detection can come very close to self-detection and non-zeroshot oracle in terms of distinguishablity*, with *smaller models being the best universal detectors*, regardless of the generator architecture or training data. For instance for GPT4 the AUC of using OPT-125M as a cross-detector is 0.90, whereas OPT 6.7B's AUC is 0.71. We then further in-

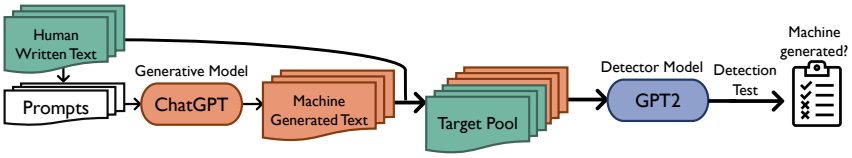


Figure 1: Overview of our methodology: We study how models can *cross-detect*, i.e. distinguish between human-written text and machine-generated text generated by another model. We create a *target pool* of both human-written and machine-generated text and feed the pool to the surrogate *detector model* to get the value of the signal we want to use (likelihood, curvature, etc.) and use this signal to test if the sequence is machine-generated or not.

vestigate some possible reasons for this phenomenon by analyzing curvature and log-likelihood of the different models, and find that larger models are more conservative in terms of the likelihood and curvature they assign to generations from other models. Smaller models, however, assign higher curvature to generations of models their size or larger, therefore they can be used to cross-detect on a broader range of models.

2 Methodology and Experimental Setup

Figure 1 shows the methodology of our work, where for a given *target pool* of sequences (with a 50%/50% composition of machine-generated/human-written text), the task is to *determine if each sequence is human-written or machine-generated* by running a detection test using the likelihood surface of a surrogate *detector model*.

Detection test. We try various detection test-based zero-shot methods that rely on the predicted token-wise conditional distributions of the generator model for detection. However, these methods were originally intended for self-detection (i.e. detecting text from a known, available generator using the model itself), whereas we test them in a cross-detection setup (i.e. using the surrogate detector model). We use four different signals for our detection tests. (1) *log-likelihood*: average token-wise log probability, with the intuition that sequences with higher log probability are more likely to be machine-generated. (2) & (3) *rank* and *log-rank* (Solaiman et al., 2019) which is the average observed rank or log-rank of the tokens based on the detector model, with the intuition that machine generated text tends to have lower rank. (4) *curvature* (Mitchell et al., 2023; Mattern et al., 2023), which uses the local-optimality of a point with respect to its neighbors (i.e. perturbations), in the likelihood surface of the detector model. The intuition is that if the likelihood of a point is much higher than most of its neighbors, then it is more likely to be machine-generated. For all these signals, the detection test places a threshold on the value and determines human-written vs. machine generated based on that. We compare results to an *Oracle*, non-zero shot baseline, which is the `openai-roberta-base` model, a classifier specifically trained by OpenAI to detect machine-generated text.

Success metric. We evaluate the success of the detector by measuring the area under the ROC curve (AUC), i.e. the false positive vs. true positive rate curve. The higher the AUC, the more distinguishing power the detection mechanism has. We use this measure as it is threshold independent and measures the true power of the method.

Models and datasets. For full details of the experimental setup refer to Appendix A. Here we discuss a brief summary. We use models ranging from 70 Million to 6.7B parameters as detectors, including the OPT, GPT, GPT-J, GPTNeo and Pythia families (Biderman et al., 2023; Zhang et al., 2022; Wang and Komatsuzaki, 2021). For our evaluations, We use a subsample of the SQuAD (Rajpurkar et al., 2016) and WritingPrompts (Fan et al., 2018) datasets, where the original dataset sequences are used as the human-written text in the target sequence pool. We then use the first 20 tokens of each human-written sequence as a prompt, and feed this to the target model, and have it generate completions for it. We report results averaged over these datasets.

3 Does cross-detection work?

In this section we present our experimental results and show that *cross-detection can perform as well as self-detection and come very close to a non-zero shot oracle baseline*. We also experiment with partially trained checkpoints of different detector models, and find that for larger models, partially trained checkpoints are better cross-detectors than fully trained ones (results are in Appendix B.5 as they are not the main focus of this paper).

3.1 Smaller Models Are Better Detectors

Figure 2 shows distinguishablity results using curvature as the test signal, where the rows are the generator models (sizing up from bottom row to top)

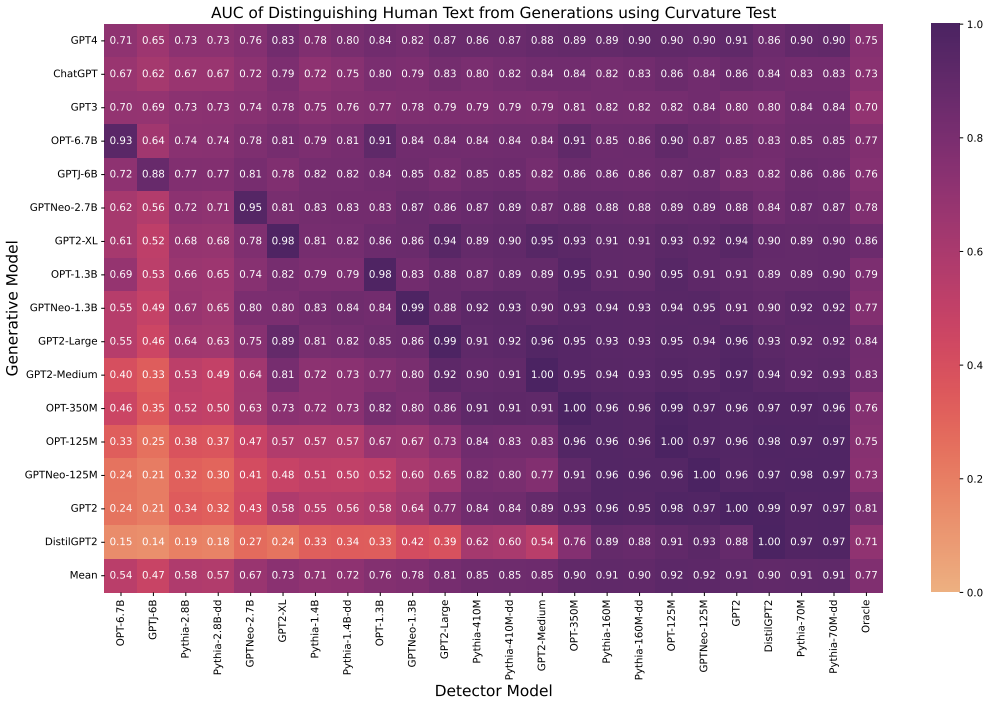
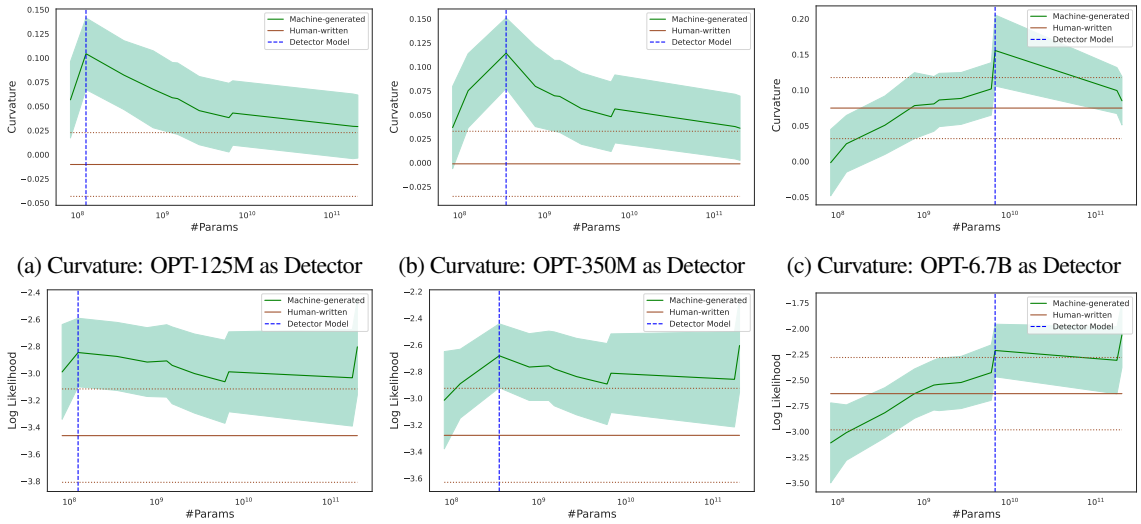


Figure 2: AUC heatmap for cross-detection, where the rows are generator models and columns are the surrogate detector models, both sorted by model size. We can see that smaller models are better detectors.



(a) Curvature: OPT-125M as Detector (b) Curvature: OPT-350M as Detector (c) Curvature: OPT-6.7B as Detector

(d) Loglikelihood: OPT-125M as Detector (e) Logliklihood: OPT-350M as Detector (f) Logliklihood: OPT-6.7B as Detector

Figure 3: Comparison of curvature and log likelihood values (mean and standard deviation) for the best universal detector (OPT-125M), a medium sized detector (OPT-350M), and a larger detector from the same family (OPT-6.7B) on generations from models of various sizes (x-axis). The 'Detector Model' line shows values for when the generator and detector are the same model. Detectors tend to show higher curvature on generations than human-written text only for generations from models of the same size or larger.

and the columns show the detector models (sizing up from right to left). Each cell shows the detection power (AUC). The last row is the mean, which is an overall metric of how good of a detector that model is. Figure 4 shows a summary of it for the other three signals, with extensive heatmaps in Appendix B.6.

We see that the bottom left has the lowest values, showing that *larger models are not good at detecting machine generated text from other models*, and they are particularly bad at it for detecting small model generations. We can also see that **smaller models are much better detectors**, as the right side of the graph has much higher AUC values. This trend holds across all the four different detection tests. Another observation is the correlations between the **dataset** and **model architecture** of the generator

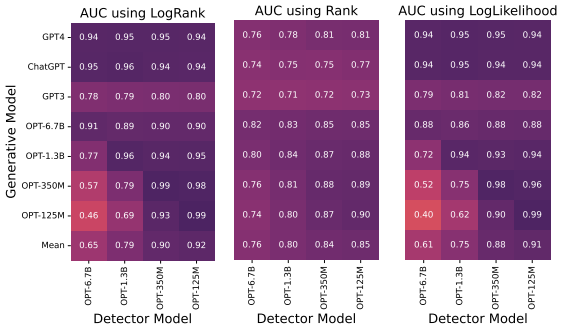


Figure 4: Summary of AUC results for signals other than curvature. We see a similar trend, with smaller models providing a better distinguishing signal.

and detector models. As the heatmap shows, models from the same *architecture family* and trained on the same/overlapping *dataset* are better at detecting their own text, compared to models from a different family. For instance, for detecting text generated by OPT-6.7B the other models from the OPT family are the best cross-detectors, with AUCs ranging from 0.89-0.87 (OPT-6.7B self-detects with AUC 0.91). The next best cross-detector is the smallest GPTNeo-125M with AUC 0.86. However, the OpenAI GPT2 model of the same size has a lower AUC of 0.84 (and overall the GPT2 family has the lowest cross-detection AUC on OPT), which we hypothesize is due to the larger gap in the training data, as the OPT and GPTNeo/GPTJ models are all trained on the Pile dataset, but GPT2 is trained on the Webtext. All in all, the difference due to the dataset/architecture differences is small as most of the dataset for all these models is comprised of web-crawled data. The right-most column is the non-zero shot oracle baseline, and as we can see cross-detection comes close to it, especially for larger models.

One noteworthy observation is that OPT-125M can detect generations from models like GPT3 and ChatGPT with relatively high AUC (0.82). However, if the intuitive approach of taking another large, "similar" model were to be taken and we were to use OPT-6.7B, we would get AUC of 0.70 and 0.67 for these models, respectively, which are both close to random (0.5).

3.2 Curvature and Loglikelihood Breakdown

We plot a breakdown of the curvature metric (Section 2) and log-likelihood values for the best universal detector (OPT-125M), a medium sized detector of the same family (OPT-350M) and a larger one from the same family (OPT-6.7B), shown in Figure 3. The y-axis is the curvature/log likelihood of the target generations under the detector models (OPT-125M, 350M or 6.7B). The x-axis is the number of parameters of the generator model.

We can see that for the smaller detector model (Figures 3a and 3d), the mean curvature and log-likelihood values for the generated text are consistently higher than the curvature for the human-written text. However, for the larger model (Figure 3c and 3f), the curvature and log-likelihood values for the machine-generated text is in most cases smaller than or around the same value as the human written text. The curvature and log-likelihood values for human written text for both graphs are stable since the text is the same and doesn't depend on the target model.

We can also see that overall the curvature and likelihood values for the larger model are higher, especially for the original text, than those of the smaller model, and the values for text generated by the other models have lower curvature and likelihood value. This shows that the larger model places higher likelihood on the human written text and fits it better. The smaller model, however, assigns lower curvature and likelihood to the human-written text compared to generations by a large gap, and the assigned values are overall lower than those of the large model. Broadly we observe that **all models respond similarly to machine generated text from other models, so long as the other model is same size or bigger.** In other words, they place high likelihood on text from larger models. However, for models smaller than themselves, they place lower likelihood and curvature. As such, smaller models are **better universal detectors**, as the size of the set of sequences they assign higher likelihood and curvature to is bigger than it is for large models, and this higher curvature is much higher than the curvature assigned to the human written text. The spikes in all the sub-figures of Figure 3 graphs are for the detector model detecting its own text.

4 Conclusion

With the increasing prevalence of LLMs it becomes crucial to differentiate between text written by humans and text generated by machines so as to avoid fake news and impersonations. As such, we set out to explore the possibilities of using existing models to detect generations from unknown sources, and distinguish them from human written text. We find that when using zero-shot detection methods, smaller models are overall better at detecting generations, and larger models are poor detectors. Our results offer hope of robust general purpose protection against LLMs used with nefarious intentions.

Limitations

Although we see high AUCs for black-box detection of machine generated text in our experiments, this does not necessarily mean that these detection methods are not avoidable, and that they can be applied to all models and achieve high performance. We present further experiments in Appendix B.7 to see the performance degradation when paraphrasing is used to avoid detectors, and find it to be not significant. However, further experiment are needed to evaluate the generalization of our findings to other architectures and setups. As LLMs continue to change and detection evasion methods become more prevalent, so must methods for detection and validation studies.

References

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *42nd IEEE Symposium on Security and Privacy*.

Mikhail J. Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding*, pages 185–200, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.

Monica Chew and Henry S. Baird. 2003. Baffletext: a human interactive proof. In *IS&T/SPIE Electronic Imaging*.

Liam Dugan, Daphne Ippolito, Arun Kirubarajan, Sherry Shi, and Chris Callison-Burch. 2022. Real or fake text?: Investigating human ability to detect boundaries between human-written and machine-generated text.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with LSTMs. In *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106, Vancouver, Canada. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. Mgtbench: Benchmarking machine-generated text detection. *arXiv preprint arXiv:2303.14822*.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

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.

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. *arXiv preprint arXiv:2303.13408*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Ji, Bernhard Scholkop, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership inference attacks against language models via neighbourhood comparison. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL Findings)*.

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*.

OpenAI. Gpt-2: 1.5b release.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395, Online. Association for Computational Linguistics.

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon Lee. 2021. TURINGBENCH: A benchmark environment for Turing test in the age of neural text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2001–2016, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2023. Ghostbuster: Detecting text ghostwritten by large language models. *arXiv preprint arXiv:2305.15047*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Hong Wang, Xuan Luo, Weizhi Wang, and Xifeng Yan. 2023. Bot or human? detecting chatgpt imposters with a single question.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852.

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.

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Association for Computational Linguistics.

A Extended Experimental Setup

A.1 Models

We want to experiment with a wide range of models, with different architectures, parameter counts and training datasets, therefore we use the following model families in our experiments: Facebook's OPT (we use the 125M, 350M, 1.3B, and 6.7B models), EleutherAI's GPT-J, GPTNeo and Pythia (Biderman et al., 2023) (we use GPTNeo-125M, GPTNeo-1.3B, GPTNeo-2.7B, GPTJ-6B and Pythia models ranging from 70M to 2.8B parameters), and OpenAI's GPT models (distilGPT, GPT2-Small, GPT2-Medium, GPT2-Large, GPT2-XL, GPT-3 and ChatGPT).

We also have experiments where we use partially trained models as detectors. For those experiments, we only use the Pythia models as they are the only ones with available, open-source partially trained checkpoints. For each Pythia models, there is also a de-duplicated version available, where the model is trained on the de-duplicated version of the data, as opposed to the original dataset. All the models we use are obtained from HuggingFace (Wolf et al., 2019).

A.2 Dataset

Evaluation dataset. We follow Mitchell et al. (2023)'s methodology for pre-processing and feeding the data. We use a subsample of the SQuAD dataset (Rajpurkar et al., 2016), where the original dataset sequences are used as the human-written text in the target sequence pool. We then use the first 20 tokens of each human-written sequence as a prompt, and feed this to the target model, and have it generate completions for it. We then use this mix of generations and human-written text to create the target pool for which we do the detection. In all cases, following the methodology from Mitchell et al. (2023), our pool consists of 300 human-written target samples, and 300 machine-generated samples, so the overall pool size is 600.

Pre-training datasets for the generative models. The EluthterAI and Facebook models (GPTJ, GPT-Neo, Pythia and OPT families) are all trained on the Pile dataset (Gao et al., 2020), a curated collection of 22 English language datasets (consisting of web-crawled data, academic articles, dialogues, etc.). As mentioned above there are two versions of each Pythia model (Biderman et al., 2023), one version is trained on Pile, the other is trained on de-duplicated Pile. The de-duplicated Pile is approximately 207B tokens in size, compared to the original Pile which contains 300B tokens. There is limited information and access to the training data of the *OpenAI* models. The GPT-2 family is reportedly trained on the WebText dataset, GPT-3 is trained on a combination of the Common Crawl [1], WebText2, books and Wikipedia, and there is not any information released about the training data of ChatGPT.

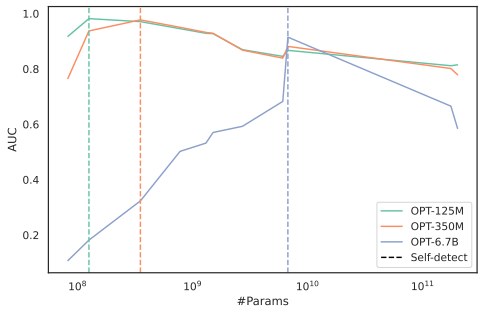


Figure 5: AUC of the three cross-detectors from Figure 3

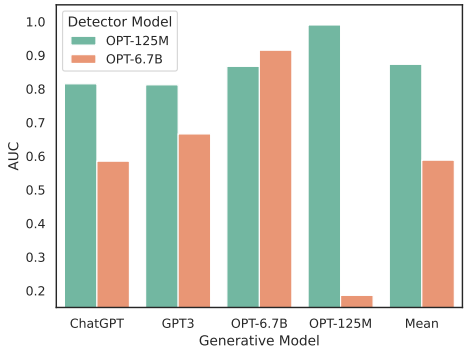


Figure 6: Summary of the cross-detection area under the ROC curve (AUC) results for a selection of generative (the 4 models over the X axis) and detector (OPT-125M and OPT-6.7B) models. We can see that the smaller OPT model is a better universal cross-detector. Full results are shown in Figure 2.

B Additional Plots and Experiments

B.1 Does neighborhood choice matter?

Our estimation of "curvature" hinges upon generating numerous perturbations (neighbors) and comparing their loss with that of a target point. Therefore, if these perturbed neighbors are not sufficiently nearby and lie in a different basin of the likelihood surface, our measure of curvature is not accurate (the closer the perturbed points are, the more accurate estimation of curvature we achieve). The perturbation method directly impacts the size and shape of the neighborhood we create. Therefore, we compare different pertur-

[1] https://commoncrawl.org

bation schemes in order to see how sensitive detectors of different sizes are to neighborhood choice.

We investigate two different methods for changing the distance of the generated perturbations: (1) we change the mask filling model size, by experimenting with *T5-Small*, *T5-Large* and *T5-3B* (Wolf et al., 2019; Raffel et al., 2020) to test the intuition that larger mask-filling models, generate semantically closer neighbors than a smaller model, we present the extended results for this in Appendix B.4. A similar analysis is also conducted in (Mitchell et al., 2023), we however, do a more extensive analysis on numerous models of different sizes and probe the curvature values. (2) We change the percentage of the tokens that get masked and replaced by the mask-filling model, as the more tokens we mask and replace, the farther the generated perturbations would be. (3) Finally, we look into how many tokens we actually need in the generated/human-written sequences to create a neighborhood and be able to accurately distinguish the texts.

B.2 Masking Percentage

Figure 7 shows the results for the experiment where we change the percentage of tokens that are masked, to produce the neighbors. In all previous experiments, we used 15% masking with mask span length of 2 tokens following the experimental setup in Mitchell et al. (2023). In this section, however, we change the percentage of the masked tokens (and we set the masking to be contiguous) to see how it affects the curvature mean and standard deviation values, and the AUCs. We can see that as the masking percentage decreases (from 90% to 2%), the AUCs and the self-detection power of models increase rather consistently. When we go to 1%, however, we see the AUC drop. If we look at Figure 7e which depicts the curvature measures for the 1% masking, we see that the curvatures overlap between machine-generated and human-written text, which we hypothesize is because our implementation does not enforce that re-sampled words must differ from the words they are replacing. Thus, for the smallest masking percentage, it is possible that some perturbations are identical to the target, which may explain reduced detection accuracy in this setting[2].

B.3 How many tokens do we need for detection?

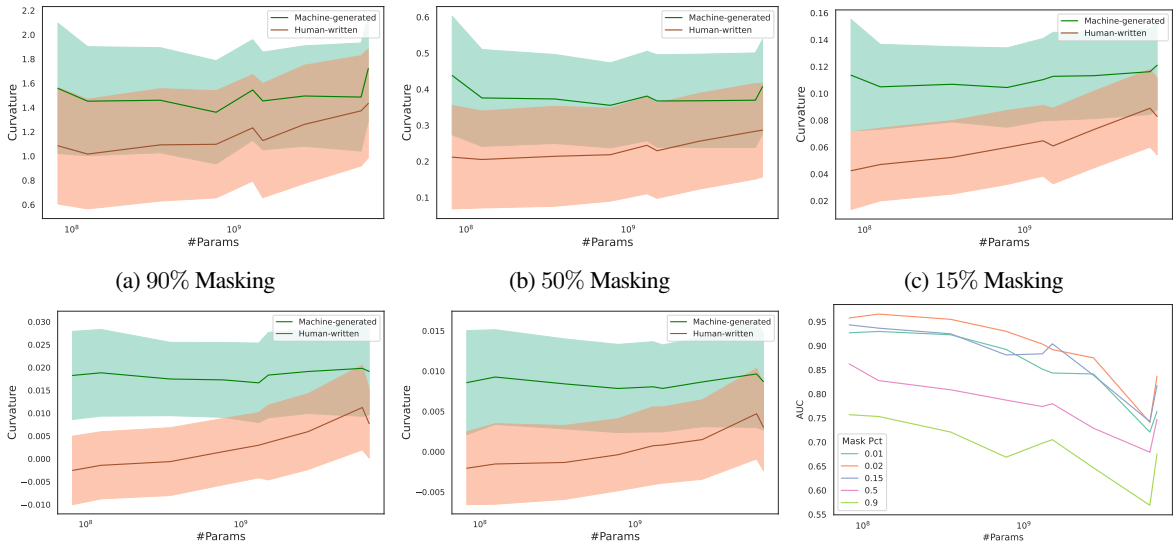
Figure 8 shows how the length of the target sequence affects the sequence's detectablity (AUC of detection), and how many tokens we need to be able to do precise detection. We compare sequences of different lengths, ranging from 10 tokens to 200, for four different models with four different parameter counts, on the SQuAD dataset. In this setup we target self-detection. We can see that the longer the sequence, the easier it is to distinguish if it is human-written or machine-generated, and 75-100 tokens seems like the point where we hit diminishing returns. We can also see that across different sequence lengths, as models get smaller, the detection power increases, as seen throughout the rest of the paper.

B.4 Ablating Mask Filling Models

Figure 12 shows the curvature numbers for each model trying to **detect its own** generations, so for each model the generator is also the detector. We experiment with three perturbation generating models, with three different sizes: (1) T5-small (60 million parameters) (2) T5-Large (770 million parameters) (3) T5-3B (3 billion parameter). The intuition behind using three model sizes is to see the effect of having a better replacement model on the measured curvatures and the detection power of the detector models.

We can see that as the masking model sizes down (going from top to the bottom subfigures), the overall curvature values for both human-written and machine-generated text increases (going from 0.2 maximum in Figure 12a to 0.6 maximum in Figure 12c), and the two sets of texts become less distinguishable. T5-Small produces low-quality (low-fluency) neighbors that are assigned lower likelihoods by the detector model, resulting in high curvature numbers for both human and machine generated text, making them indistinguishable. As we improve the mask filling model, however, the generated neighbors become of higher quality (and semantically closer to the target point), thereby creating a more accurate estimate of the curvature and providing better distinguishablity, as shown by the AUC numbers in Figure 12d.

[2]Its noteworthy that the slight discrepancy between the results for 15% masking in this section and the previous section is that there, the mask span length was 2 so the masked portion of the sequence is not contiguous. In this experiment, however, we use contiguous masking.



(a) 90% Masking (b) 50% Masking (c) 15% Masking

(d) 2% Masking (e) 1% Masking (f) AUCs for different masking pctgs.

Figure 7: The effect of changing the masking percentage on curvature values and self-detection power of different models with different sizes (AUC).

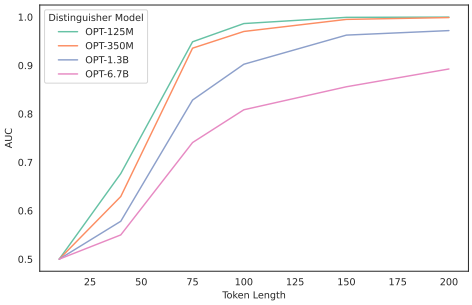


Figure 8: Detectability as a function of candidate utterance length. As expected, longer utterances are more cross-detectable – though it's worth noting that utterances as short as 60 tokens long are still cross-detectable with relatively high accuracy.

B.5 Partially Trained Models are Better Detectors

We take different training checkpoints of the Pythia models (Biderman et al., 2023) at different steps (steps $1k$, $5k$, $10k$, $50k$, $100k$ and $143k$) with different sizes (2.8B, 410M, and 70M), and use them as detectors of generations from the 4 target models. Figure 13 shows the results for this experiment (Figures 9 and 10 show entire heatmaps of this experiment). For each model we can see that **the final checkpoint is consistently the worst one in terms of machine-generated text detection**, and it is one of the middle checkpoints that has the best performance. Our hypothesis for this is similar to that of Section 3, where we believe that partially trained models have not yet fit to the training data tightly (and have a smoother surface), so they over claim other models' generations as their own, whereas the longer a model is trained, the sequences it ranks higher as its own narrow down.

B.6 Extensive Heatmaps

We provide the full heatmaps from experiments of Section 3 here, to provide a detailed breakdown. Figures 2 and 13 (full heatmap is Fig. 9 in Appendix) show the AUC of cross-detection for different models. Figures 11 and 10 in Appendix show how close each detector comes, in terms of AUC, to self-detection. Figures 14, 15 and 16 show the full heatmaps for signals other than curvature.

We provide a summary of Figure 2 in Figure 6, where we have presented the numbers from the best overall detector with mean AUC of 0.92 (OPT-125M) and the biggest model of the same family, OPT-6.7B with average AUC of 0.46.

B.7 Detection performance under a paraphrase attack

We present additional results where we perform an adaptive paraphrasing attack (Sadasivan et al., 2023) on the machine generated text and then evaluate cross-detection performance. We conducted experiments on the SQuAD test set. You can find the results in Tables 1 and 2.

We can see that paraphrasing machine-generated text does reduce detection performance to some degree. However, the detection accuracy after the paraphrase attack is high enough for detection to still

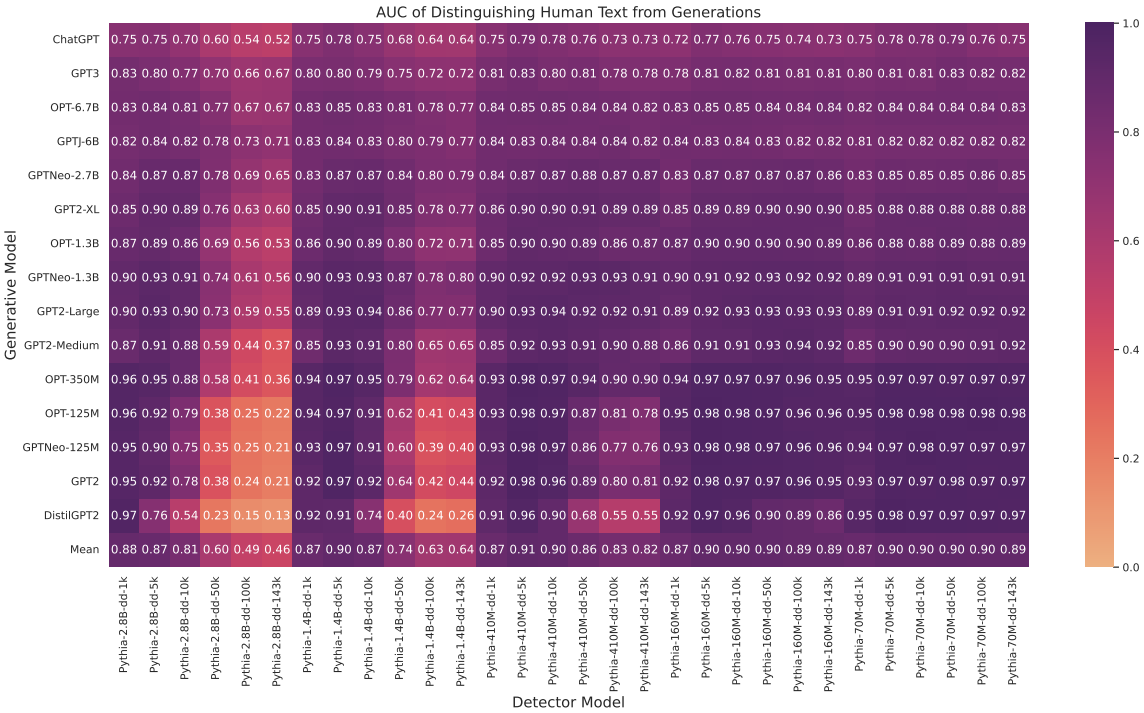


Figure 9: AUC heatmap for cross-detection, where the rows are generative models and columns are the surrogate detector models from the Pythia family, at different training step checkpoints ($1k$, $5k$, $10k$, $50k$, $100k$ and $143k$), both sorted by model size. We can see that partially trained models are better detectors.

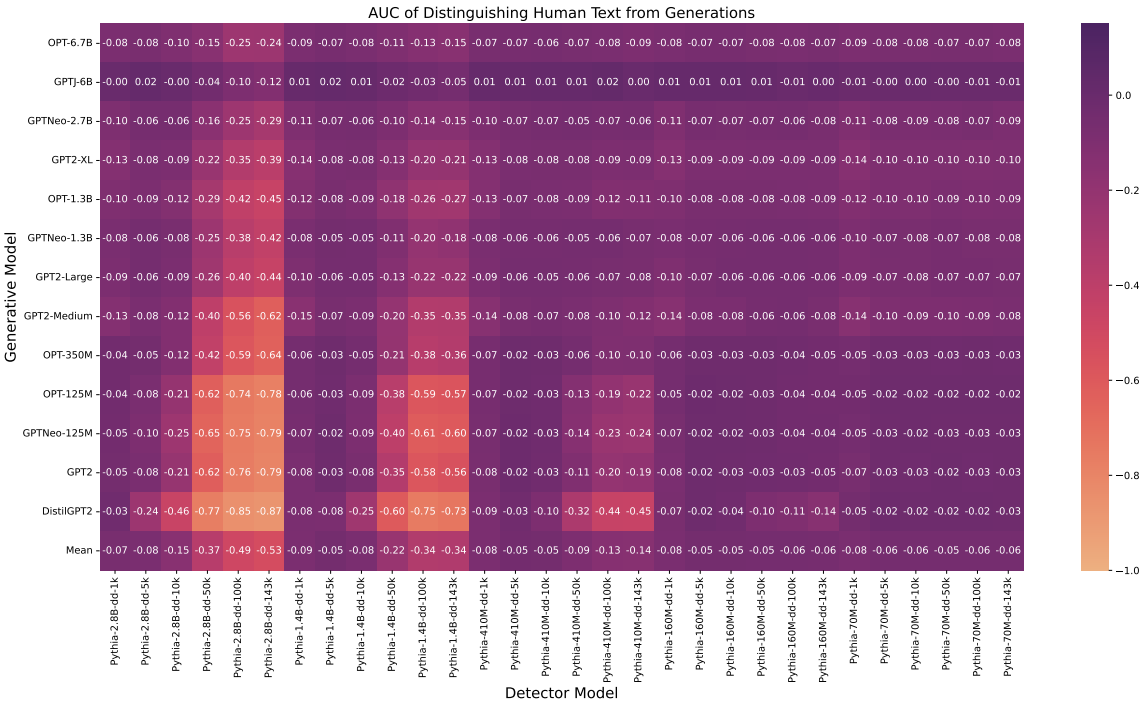


Figure 10: AUC difference between self-detection and cross-detection heatmap (to better see how close cross-detection comes to self detection), here the rows are generative models and columns are the surrogate detector models from the Pythia family, at different training step checkpoints ($1k$, $5k$, $10k$, $50k$, $100k$ and $143k$), both sorted by model size. This plot is basically Figure 9, where each cell in a row is subtracted by the self-detection AUC for that row.

be practically useful (the mean AUC for OPT 125M goes from 0.946 without paraphrase to 0.84 with paraphrase). While at first this might seem surprising, in a sense, detecting the outputs of the paraphrase system

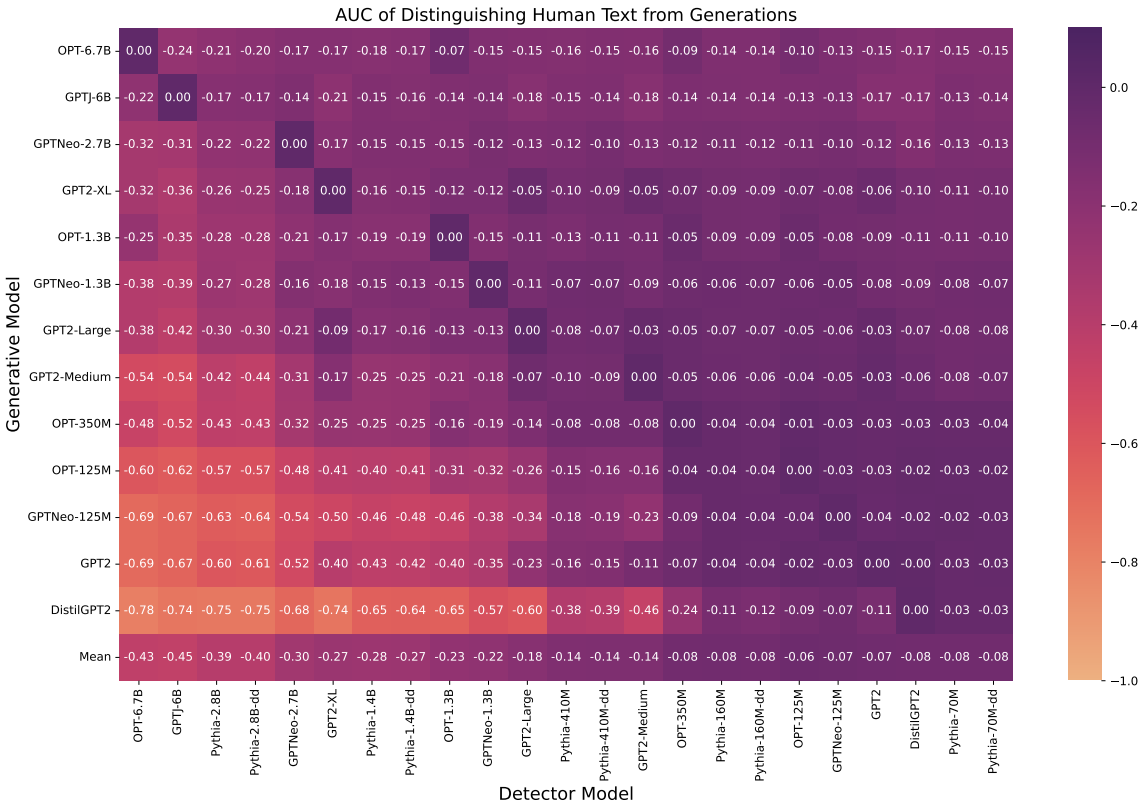


Figure 11: AUC difference between self-detection and cross-detection heatmap (to better see how close cross-detection comes to self detection), where the rows are generative models and columns are the surrogate detector models, both sorted by model size. This plot is basically Figure 2, where each cell in a row is subtracted by the self-detection AUC for that row.

(we use T5, according to the (Sadasivan et al., 2023)) is just another type of cross-detection: the paraphrase system is itself a language model. We've already seen in other experiments that small detectors are capable of accurate detection of outputs from completely unrelated language models – the paraphrase model seems to be no different, if somewhat further afield with respect to its training data and architecture (encoder-decoder). Finally, the trend of smaller models being better detectors holds up even after paraphrasing.

C Related Work

The problem of machine-generated text detection has already been studied for multiple years using a variety of different approaches (Ippolito et al., 2020; Jawahar et al., 2020; Uchendu et al., 2020, 2021): Both Gehrmann et al. (2019) and Dugan et al. (2022) have found that humans generally struggle to distinguish between human- and machine-generated text, thereby motivating the development of automatic solutions. Among those, some methods aim to detect machine-generated text by training a classifier in a supervised manner (Bakhtin et al., 2019; Uchendu et al., 2020), while others perform detection in a zero-shot manner (Solaiman et al., 2019; Ippolito et al., 2020). There is also a line of work that relies on bot detection through question answering (Wang et al., 2023; Chew and Baird, 2003), which is outside the scope of this paper.

Most recently, Mitchell et al. (2023) introduced the zero-shot method DetectGPT, which is based on the hypothesis that texts generated from a LLM lie on local maxima, and therefore negative curvature, of the model's probability distribution. Other strategies have been proposed to enable the detection of machine-generated text in the wild. Particularly through efforts on the side of the LLM provider, more powerful detection methods can be devised. One such method is watermarking, which injects algorithmically detectable patterns into the released text while ideally preserving the quality and diversity of language model outputs. Watermarks for natural language have already been proposed by Atallah et al. (2001) and have since been adapted for outputs of neural language models (Fang et al., 2017; Ziegler et al., 2019). Notable recent attempts for transformer based language models include work by Abdelnabi and Fritz (2021), who propose an adversarial watermarking transformer (AWT). While this watermarking method

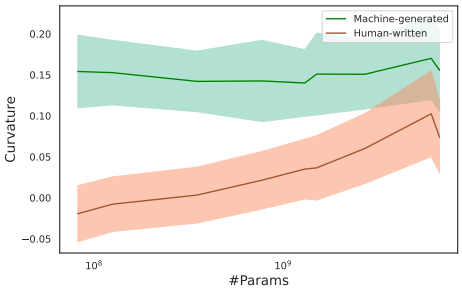
Table 1: Detection power w/o using a praphrasing attack to avoid detection.

Generator/Distinguisher	OPT-6.7b	OPT-1.3b	OPT-350m	OPT-125m
OPT-6.7b	0.915	0.888	0.881	0.867
OPT-1.3b	0.565	0.978	0.937	0.931
OPT-350m	0.320	0.780	1.000	0.989
OPT-125m	0.186	0.588	0.960	0.999
mean	0.496	0.808	0.944	0.946

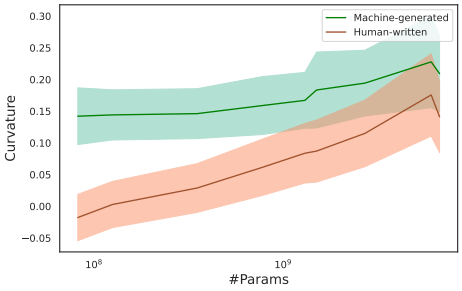
Table 2: Detection power with using a praphrasing attack to avoid detection.

Generator/Distinguisher	OPT-6.7b	OPT-1.3b	OPT-350m	OPT-125m
OPT-6.7b	0.752	0.730	0.677	0.698
OPT-1.3b	0.458	0.879	0.756	0.789
OPT-350m	0.239	0.521	0.954	0.895
OPT-125m	0.131	0.409	0.811	0.978
mean	0.395	0.635	0.800	0.840

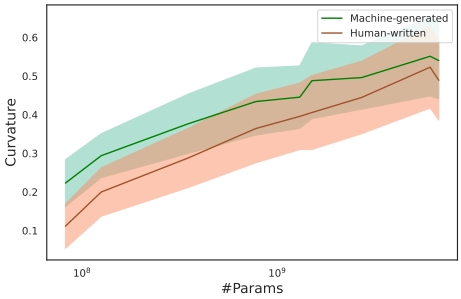
is dependent on the model architecture, Kirchenbauer et al. (2023) propose a watermark that can be applied to texts generated by any common autoregressive language model. As a strategy more reliable than watermarking, Krishna et al. (2023) suggest a retrieval-based approach: By storing all model outputs in a database, LLM providers can verify whether a given text was previously generated by their language model. In practice, this would however require storage of large amounts of data and highly efficient retrieval techniques in order to provide fast responses as the number of generated texts grows.



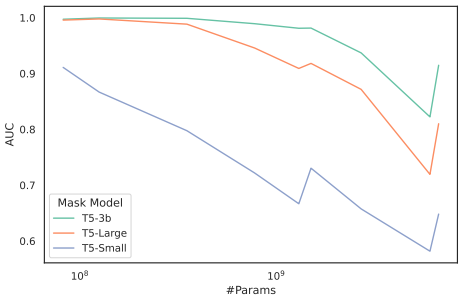
(a) T5-3B



(b) T5-Large

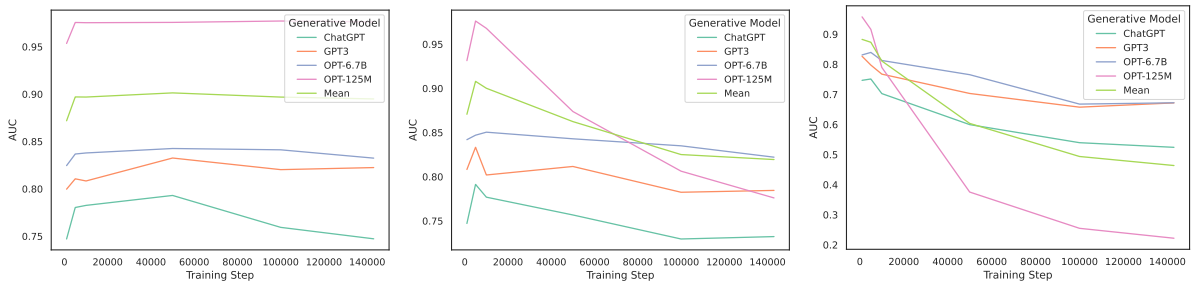


(c) T5-Small



(d) AUCs for different perturbation (masking) models

Figure 12: The effect of changing the perturbation (masking) model on curvature values and self-detection power of different models with different sizes (AUC).



(a) Pythia 70M (b) Pythia 410M (c) Pythia 2.8B

Figure 13: Summary of the results for cross-detection power of different detector models trained for different number of steps. Each subfigure shows a different detector model, and the x-axis shows the training step for the checkpoint used as a detector. The results for all 15 generator models are shown in Figure 9.

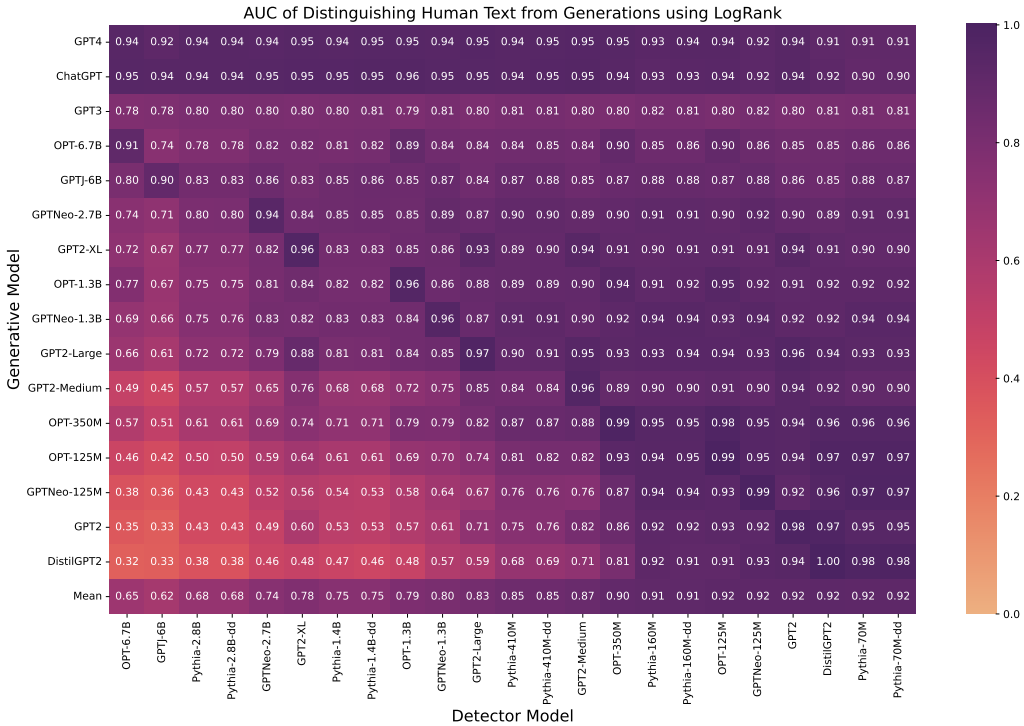


Figure 14: AUC heatmap for cross-detection, where the rows are generator models and columns are the surrogate detector models, both sorted by model size. We can see that smaller models are better detectors and larger models are the worst models in terms of detection power. The signal used here is Log Rank.

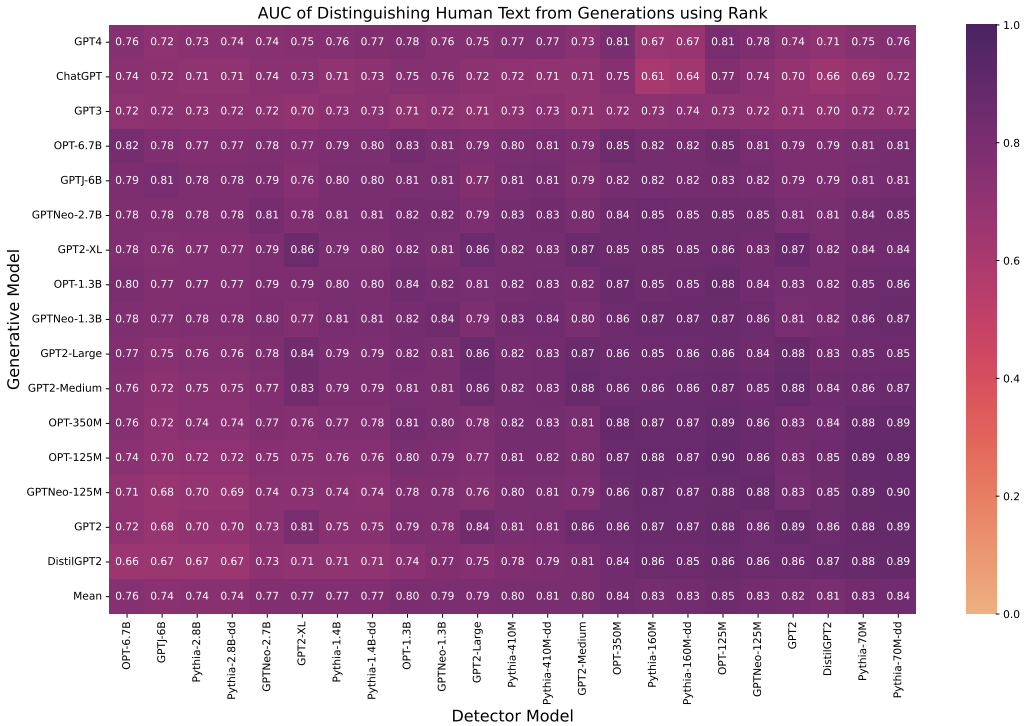


Figure 15: AUC heatmap for cross-detection, where the rows are generator models and columns are the surrogate detector models, both sorted by model size. We can see that smaller models are better detectors and larger models are the worst models in terms of detection power. The signal used here is Rank.

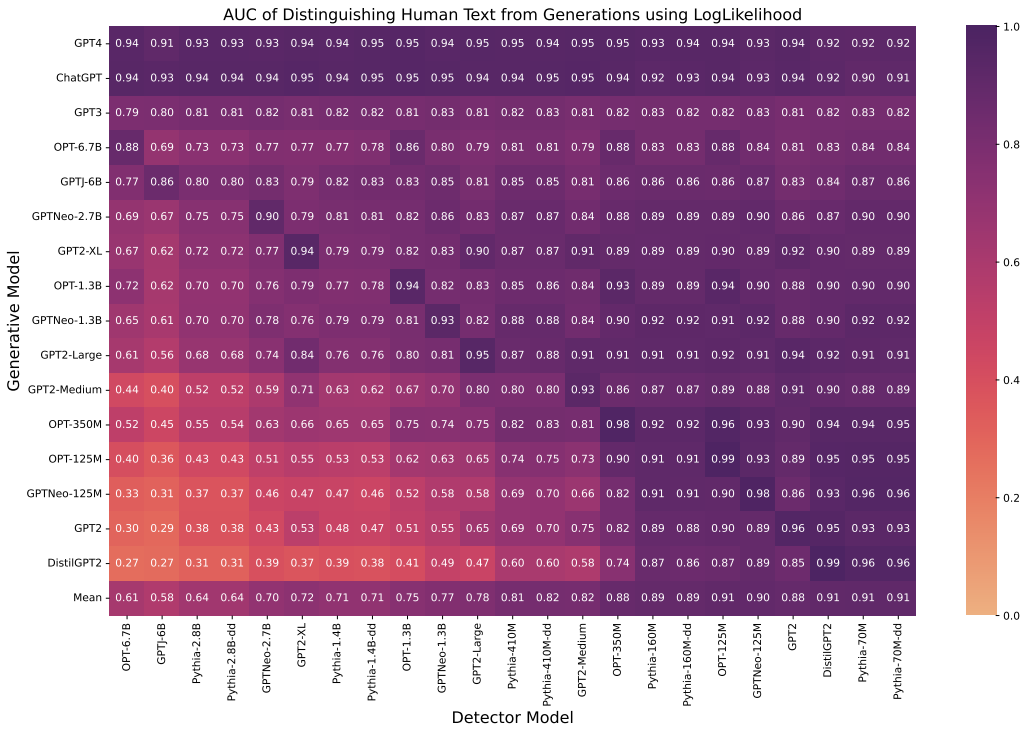


Figure 16: AUC heatmap for cross-detection, where the rows are generator models and columns are the surrogate detector models, both sorted by model size. We can see that smaller models are better detectors and larger models are the worst models in terms of detection power. The signal used here is Loglikelihood.