F Detector Details

F.1 Detectors

In this section we provide a detailed description of all detectors used in the evaluations of the RAID dataset.

RoBERTa (GPT2) (Solaiman et al., 2019) This detector[16] is a RoBERTa model (Liu et al., 2019) fine-tuned on the GPT2 output dataset. This dataset consists of outputs from GPT2 in open domain settings with three different decoding strategies: greedy decoding, top-k=50, and fully random sampling and has been a baseline inclusion for many years. We use both the base and large size of this model in our comparisons.

RoBERTa (ChatGPT) (Guo et al., 2023) This detector is a RoBERTa-base model (Liu et al., 2019) fine-tuned on the HC3 dataset. HC3 consists of roughly 27,000 questions paired with both human and ChatGPT answers in various domains such as reddit, medicine, finance, and law. We download and query the detector via HuggingFace datasets with the unique identifier `Hello-SimpleAI/chatgpt-detector-roberta`

RADAR (Hu et al., 2023) This detector is a fine-tuned version of Vicuna 7B (which itself is a fine-tune of LLaMA 7B). It was trained in a generative adversarial setting alongside a paraphrase model. The paraphraser was trained specifically to fool the detector and the detector was trained to accurately detect generations from the paraphraser, human-text from the WebText dataset, and outputs from the original language model. We download and query this detector from HuggingFace with the unique identifier `TrustSafeAI/RADAR-Vicuna-7B`

GLTR (Gehrmann et al., 2019) Originally intended as an interface to help humans better detect generated text, GLTR has become a standard baseline in robustness studies of detector abilities. GLTR evaluates the likelihood of text according to a language model and bins tokens according to their likelihoods and uses these bins as features for detection. We use the default settings from the GLTR repository[17] namely our cutoff set at rank=10 and the language model set to GPT2 small.

FastDetectGPT (Bao et al., 2023) This detector is an improvement on the original DetectGPT (Mitchell et al., 2023)—speeding up inference by 340x without any reduction in accuracy. For the scoring model we use the repository default of GPT-Neo-2.7B and for the reference model we again use the default of GPT-J-7B. Since neither of these models were used to generate continuations in our dataset, we felt that this was a reasonable choice.

Binoculars (Hans et al., 2024) This detector uses perplexity divided by cross-entropy between two very similar language models as the metric for detection. In our implementation we use the code from the official GitHub repository and calculate perplexity using the default models from the repository, namely Falcon 7B and Falcon 7B Instruct (Almazrouei et al., 2023). Much like FastDetectGPT, since neither of these models were used to generate continuations, we believed this made for a fair comparison.

LLMDet (Wu et al., 2023) This detector computes the proxy-perplexity of the input text from 10 different small language models and uses these features for detection. The proxy-perplexity is an approximation of the true perplexity calculated by repeatedly sampling n-grams from models rather than by actually running them. Of the models used, none of them were used for generations in our dataset, thus this was a fair comparison.

GPTZero (Tian and Cui, 2023) GPTZero is a closed-source commercial detector released in January 2023 and was among the first to gain widespread media attention for their detection-as-a-service business model. We queried the detector on January 24th 2024 using the v2 API[18] and threshold on the `completely_generated_prob` field.

Originality Originality is a closed-source commercial detector released in November 2022 and was the first company to adopt the detection-as-a-service business model. We queried this detector from January 24th to 25th through the v1 API[19] and threshold on the 'score' field in the output JSON. For the Czech and German news domains, we specifically query the multilingual "version 3" of the detector.

Winston This detector is the closed-source commercial detector that claims the highest accuracy

[16] We download the model hosted by OpenAI from the following link https://openaipublic.azureedge.net/gpt-2/detector-models/v1/detector-large.pt

[17] https://github.com/HendrikStrobelt/detecting-fake-text

[18] https://api.gptzero.me/v2/predict/text

[19] https://api.originality.ai/api/v1/scan/ai

out of any detector (99.98%). We query this model through the v1 API[20] and specifically set the input language to German when we detect that our input text is in German. Unfortunately, since winston does not support detection in Czech, we use English as the input language for the Czech news domain. We once again threshold on the 'score' field in the output JSON.

ZeroGPT This detector is the final commercial detector. We run this as it is the only detector besides GPTZero that has been evaluated in another benchmark paper. We query the API at the following link[21] and use the 'isHuman' return field as the classifier output.

F.2 Thresholds

In order to find the thresholds that achieve a fixed false positive rates we had to implement some basic search procedure. We searched our thresholds in a linear fashion: We start at the threshold corresponding to the mean score of human data (50% FPR) and approach the desired false positive rate by iteratively incrementing or decrementing the threshold. If we overshoot the target fpr we divide our step size in half and flip the sign. We continue to do this until the false positive rate is within $\epsilon = 0.0005$ of the desired false positive rate or until 50 iterations are reached. If 50 iterations are reached without convergence then we select the threshold corresponding to the FPR that is closest to the target while still being less than the target. If there is no such threshold then we note that the detector could not reach the target FPR and simply choose the value closest to and greater than the target. In Table 13 we list the thresholds our algorithm found for each detector along with the exact false positive rates that these thresholds allow.

G Extended Figures and Tables

G.1 Dataset Statistics and Evaluations

In Figures 10 and 11 we report extended statistics about the dataset. We see some interesting trends here, namely that human-written text is still significantly less likely than machine-generated text according to LLaMA 2 7B and that it is also the least repetitive. These results push back on claims that language models are approaching human-level performance and therefore detection is unreasonably difficult.

[20] https://api.gowinston.ai/functions/v1/predict

[21] https://api.zerogpt.com/api/detect/detectText

G.2 Extended Heatmaps

In Figure 12 we show the extended heatmaps from Figure 6 in the main paper. We see that the trend holds that RoBERTa GPT2 is significantly better on GPT2 generations and that RADAR is uncharacteristically bad on IMDb Movie Reviews.

G.3 Model Performance vs. Domain

In Table 14 we report the results of our 12 detectors across all different domains of generated text. We see that the metric-based methods such as Binoculars and FastDetectGPT generalize surprisingly well across domains. We also see that in general detectors perform well but occasionally perform surprisingly poorly.

G.4 Detector Accuracy vs. Decoding Strategy

In Table 5 we report the results of our evaluation broken up by category of model and by decoding strategy. Much like in Figure 5 from the main paper, we see that certain combinations of decoding strategies and models can cause detector accuracy to plummet unexpectedly. This raises serious concerns for the robust deployment of detectors.

G.5 Detector Accuracy vs. Adversarial Attack

In Table 16 we report the full version of the results from Table 6 in the main paper. We see similar trends, namely that certain adversarial attacks work better on certain detectors and that occasionally adversarial attacks actually improve detector performance rather than harm it. One notable inclusion here is the zero-width space attack, which seems to either cause detectors to assign all positive or all negative labels. Future work should investigate this phenomenon.

H Example Generations

In Table 17 and 18 we provide example outputs for each generative model and adversarial attack. We see that different generative models have significantly differing styles, underscoring the difficulty of the detection problem.

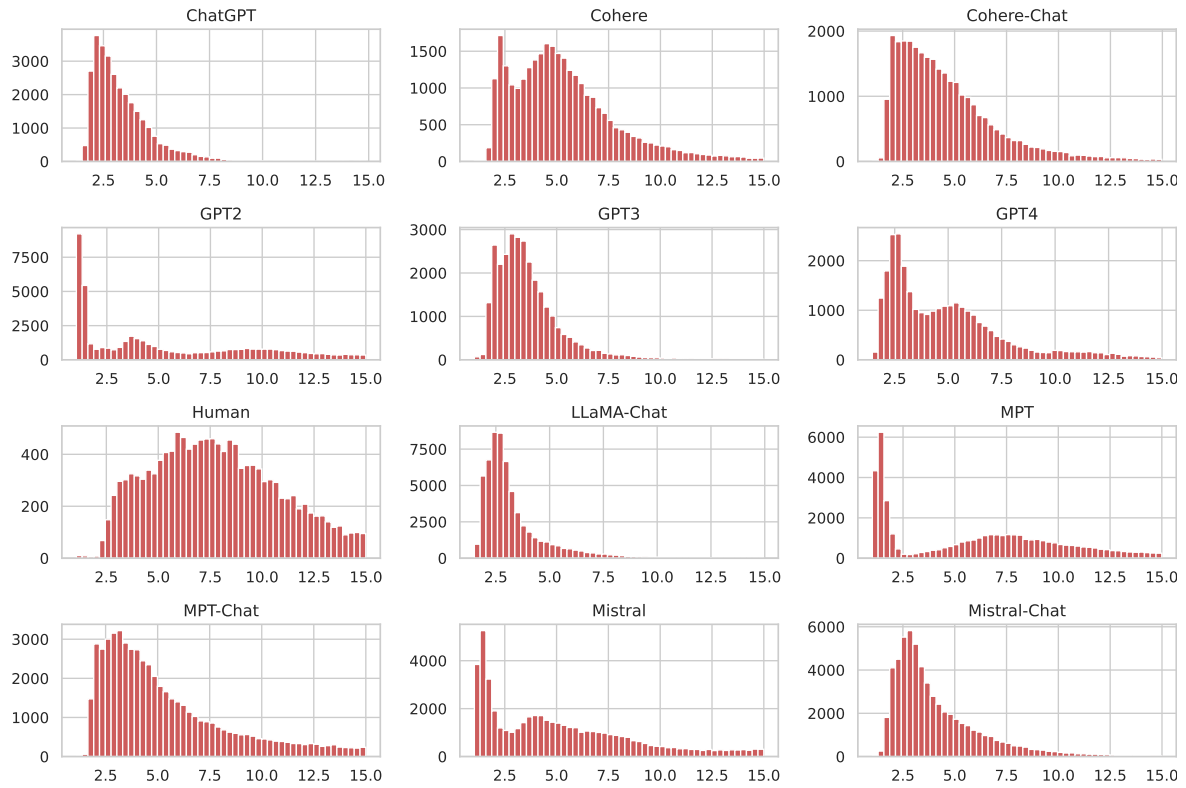

Figure 10: Histograms of perplexity according to LLaMA 2 7B per generative model in the dataset. We see that there is still a significant difference between human-written and machine-generated text with respect to perplexity.

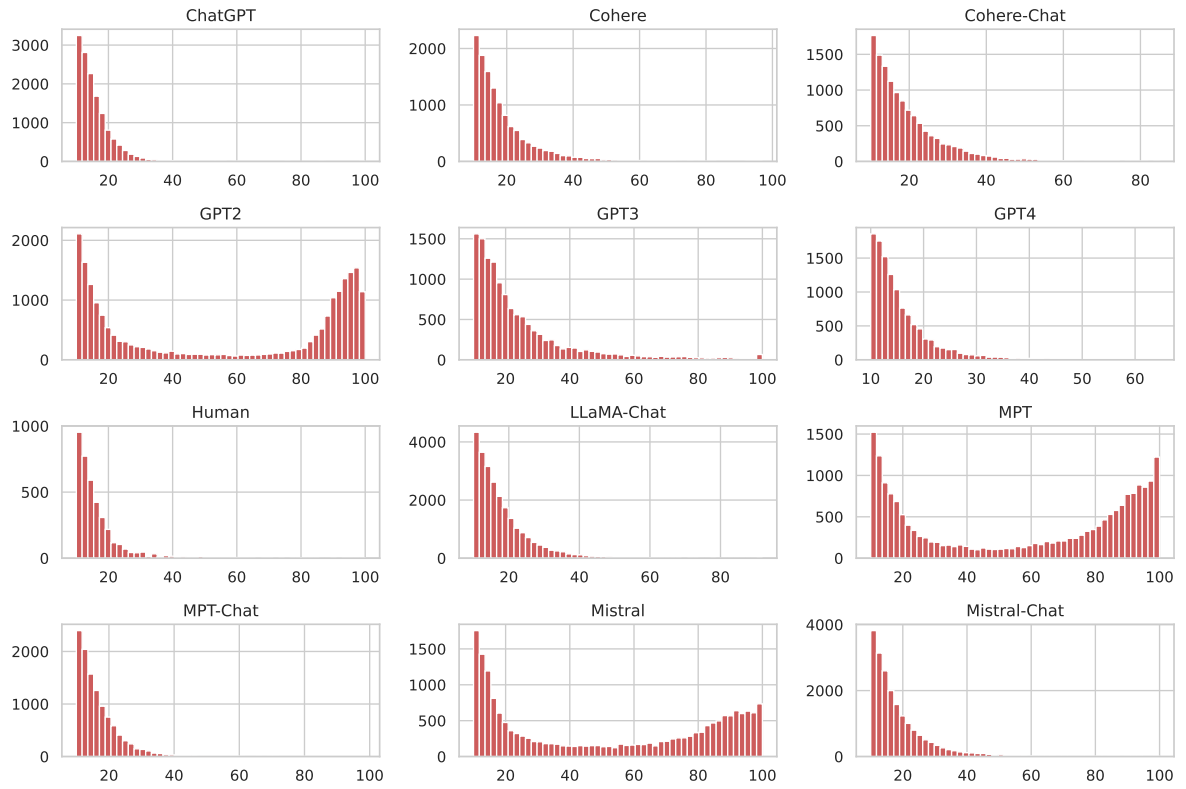

Figure 11: Histograms of SelfBLEU (Zhu et al., 2018) per generative model in the dataset. We see that continuation models tend to be more repetitive than chat models and that human-written text is by far the least repetitive.