Detecting ChatGPT Text

Stanford CS224N Custom Project

Julia Park
Department of Computer Science
Stanford University
julpark@stanford.edu

Armaan Rashid
Department of Symbolic Systems
Stanford University
armaanrashid@stanford.edu

This template is built on NeurIPS 2019 template[1] and provided for your convenience.

1 Key Information to include

- External collaborators (if you have any): N/A
- Mentor (custom project only): We have no mentor yet, however are currently trying to get in contact with Ph. D student Eric Mitchell (the author on the research paper analyzed here).
- Sharing project: N/A

2 Research paper summary (max 2 pages)

Title	DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature
Year	2023
URL	https://arxiv.org/abs/2301.11305v1

Table 1: Bibliographical information for research paper.

Background. The authors identify detection of large language model (LLM) generated text as an increasingly important problem, since popular LLMs like PaLM and OpenAI's GPT-3 and ChatGPT are producing human-like text that is often difficult to distinguish from genuine text, even to human evaluators. This similarity can cause problems in sectors like education, as teachers may no longer be able to accurately assess student learning. The paper creates a new method, using the probability distributions internal to LLMs, to detect AI-generated texts across domains.

Summary of contributions. The researchers develop a new model called DetectGPT, which uses a zero-shot approach to detecting AI-generated text. This model is especially powerful because it does not require training a separate classifier or curating a dataset of generated vs human writing. DetectGPT is shown to perform better than pre-existing zero-shot methods for identifying machine-generate passages.

Their model exploits the central feature of how LLMs produce text, which is to choose the next word with the highest probability given the words that have come before. Because every word chosen in an AI-generated text has the highest probability, the researchers hypothesize that small perturbations in an AI-generated passage **almost always** leads to a passage that has a lower overall probability of being produced by the model (as judged by the model itself). This is an intuitive assumption: since the non-perturbed AI passage generated by some LLM has each word chosen with the highest probability, any perturbations in this passage will have, as a whole, a comparatively lower (log) probability. In the authors' terms, this means that the (log) probability of an AI-generated text passage is always at a

[1] https://www.overleaf.com/latex/templates/neurips-2019/tprktwxmqmgk

local maximum of a log-probability function which is taken over passages nearby to the candidate passage.

The crucial assumption the researchers make is that, by contrast, human passages, which are obviously not generated by choosing individual words with the highest probability, have a considerably more irregular log-probability function, where that function is taken over passages near the candidate passage. Therefore, slightly perturbed versions of human passages will, when taken in and assessed for probability by an LLM, have potentially higher or lower log probabilities than the candidate with more randomness in the distribution over the perturbed passages, instead of a consistently lower log probability as the passage becomes more perturbed.

To test this hypothesis, the authors use the model T5-3B to generate 100 **reasonable** (i.e. preserving overall meaning) perturbations of a given candidate passage, and then assess the log probability of the candidate passages and the perturbed passages using the LLM which is suspected of generating the candidate. The discrepancy in the probabilities between the candidate and perturbed passages is measured with the function

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

where p_θ is the probability output by the source model, x and $\tilde{x}$ are the candidate and perturbed passages respectively, and q is the function which is used to perturb examples. The researchers provide a quick mathematical proof showing that this function does indeed approximate the curvature of the log probability function taken over a candidate and lexically close perturbed passages, which, to repeat from earlier, is hypothesized to always be negative for an AI-generated passage, and irregular for a human passage.

Perturbing examples, measuring the perturbation discrepancy described above, and then judging a passage to be machine generated or not if the discrepancy is greater than some ϵ threshold, the authors indeed find that over three different standard datasets (XSum for fake news, SQuAD Wikipedia paragraphs as a stand in for academic papers, and stories from Reddit WritingPrompts as a stand in for creative writing) this method has state of the art effectiveness for detecting machine generated passages. Using the AUROC metric, which measures if a classifier correctly ranks a random AI-generated passage as more likely to be AI-generated than a random human passage, this method outperforms similar zero-shot detection methods across all datasets and when attempting to detect text from a range of LLMs (like GPT-2, GPT-J, Neo, etc). This also holds when the performance is measured over passages that are more heavily perturbed: all methods become worse at detection but DetectGPT still outperforms others, cementing its status as the most effective zero-shot detection method currently available.

Limitations and discussion. The authors do a great job of controlling for different hyperparameters (they test over a sweep of different span lengths for perturbation and overall percentage of passage that is perturbed) to see if DetectGPT still outperforms other zero-shot models. They also do one comparison, for GPT-3, between DetectGPT and the supervised-learning-based detector (i.e. a detector trained on labeled human and labeled AI examples) RoBERTa, which is trained for detection of GPT-3 text, still finding DetectGPT competitive with the best RoBERTa model's results. It is unfortunate, however, that for the other models (GPT-2, GPT-J, OPT-2.7, NeoX, Neo-2.7) that comparisons aren't made with detectors trained on labeled data, making it hard to generalize those results past the realm of zero-shot text detection. The chosen datasets also reflect a reasonable variety of natural language data.

Finally, though the authors allude to this problem for future research (which our project aims to conduct) this paper largely assumes 'whitebox' access to the probability distributions output by the LLM, which aren't available in many cases with the LLMs that are actually being widely used (and therefore in most dire need of detection), ChatGPT being the most pointed example given its sudden and widespread popularity beyond the AI/technology community. The researchers allude to possible methods for querying other similar language models to the one that's trying to be detected, in order to perform detection. They do perform experiments like this, trying to detect (for example) GPT-2 text by querying GPT-J for probabilities, predictably finding that such cross-detection attempts perform worse. They don't attempt to query **multiple** similar models and compose the probabilities in some way to generate text.

Why this paper? With the mass adoption of ChatGPT, LLM text detection is rapidly becoming necessary, especially in fields like education where ChatGPT can easily and obviously be used for plagiarism. As LLMs get better and better, it is also likely that human beings will not be able to evaluate AI-generated vs human text better than chance, as the researchers point out. This paper was chosen in this context, since the mass use of LLMs is apparently hitting an inflection point.

Wider research context. The authors point out that previous efforts to contribute to the detection of LLM-generated text have relied on one of two techniques. For one, some researchers have tried creating a deep neural network classifier which trains on LLM-generated and human language examples to classify new text one way or the other, but the drawback of this approach is that the model is likely to overfit the topics that were trained on and be less effective on classifying language from other areas. Second, a zero-shot approach to detection is used, where instead of training on different examples of AI and human text, a model is developed that out of the gate performs detection, using different methods. The researchers' main contribution is by creating a new model along the second of these axes, which according to their results is the SOTA among these kinds of zero-shot models, at least on common English language datasets.

3 Project description (1-2 pages)

Goal. The goal to expand the zero-shot machine detection model described in Mitchell et al. to language models that do not share their probability distributions. Specifically, we seek to find out if the model can be effectively applied to ChatGPT, which currently does not share its probability distribution, as OpenAI has not made its API public.

To put it in other words, we are trying to apply DetectGPT to cases where the *detection* model, the model whose text we are trying to identify, differs from the *query* model(s), the model(s) whose parameters we are accessing to perform detection.

We hope to effectively classify ChatGPT texts using the method in Mitchell et al. This involves perturbing a candidate passage (which may be LLM- or human-generated), then calculating the perturbation discrepancy (described above!) between the candidate and the perturbed passages. In our case, since we can't access the probabilities of ChatGPT, we will average over the probabilities that other similar LLMs output for these passages to measure perturbation discrepancy.

Task. When given an input of text around the length of a paragraph, the model will perform binary classification, deeming the text to be from ChatGPT or not.

Data. In order to make the results comparable with those in the original DetectGPT paper, we will utilize the datasets used there: selections from SQuAD, XSum, and WritingPrompts. If time permits, though, we want to also test this detection on other real-life datasets where ChatGPT has already been widely used: Stack Overflow comes to mind as a prominent example, since the website banned ChatGPT as too many users were using the LLM to generate answers for Stack Overflow questions.

Methods. Since ChatGPT doesn't share its probability distribution, we will attempt the technique suggested in Mitchell et al. of taking the probability distributions from other LLMs and composing them together to perform detection. What this means is essentially that we will average the perturbation discrepancy generated by each of the query models. Beyond this the same methods will be used as in that paper: perturbing examples using T5-3B, calculating perturbation discrepancy, and then thresholding. More detail on this method is described above.

If time permits, we can also train how much weight our model should give to the probabilities returned by each query model. This would require curating a labeled dataset to train/test on (and thus remove one benefit of the model from Mitchell et. al), but we are curious to see whether a model using learned weighted averages would perform significantly better at detecting ChatGPT texts than the model using an unweighted average. This effort could also help us recognize which query models contribute the most to detecting ChatGPT and improve our selection of query models for our zero-shot detection model as well.

Baselines. The baseline method that will be used is the one described above, perturbing, calculating the perturbation discrepancies for multiple models and averaging, and then thresholding to perform

binary classification. In particular our baseline will be to just start with using GPT-3.5 as a singular query model on potentially ChatGPT-generated text, since that is the most similar model to ChatGPT (given the limited information OpenAI has released about ChatGPT), using the hyperparameters that were found to be most effective in the DetectGPT paper originally (perturbing 100 examples, with perturbation span of 2, and a 15% mask rate). From there we will explore different models (and combining different models) for querying and potentially sweep across hyperparameters for further experiments.

Evaluation. Just as described earlier in the paper review, AUROC will be used to evaluate our DetectGPT variant on the XSum, SQuAD and WritingPrompts datasets, and we will compare with the AUROC scores achieved by DetectGPT in the original paper on these sets (an average 0.97, 0.92, and 0.97 respectively, as averaged over different models), to see if we can achieve comparable performance.