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

External collaborator: Eric Mitchell

Abstract

The detection of large language model (LLM) generated text has become increasingly important problem, since popular LLMs like OpenAI's GPT-3 and ChatGPT are producing human-like text that is difficult to distinguish from natural language, even to human evaluators. Recent research such as Mitchell et al. (2023) [1] have developed a zero-shot method for identifying GPT-3 text using the model's internal probability distributions. However, many models like ChatGPT do not release their internal probability distributions to the public. We extend [1]'s work by identifying ChatGPT-generated text even without access to the model's internal probability distributions. We find from preliminary experiments that querying a similar model to ChatGPT, OpenAI's GPT-3.5, surprisingly underperforms, in some cases being worse than random chance. This indicates a need to consider some of the unique aspects of ChatGPT's model design in running further experiments.

1 Approach

Our approach extends Mitchell et. al's DetectGPT [1], a zero-shot method for AI text detection. DetectGPT proceeds from the assumption that since LLMs generate text by sampling from their probability distribution, an LLM-candidate passage will always have a higher log probability (according to that LLM) than perturbed versions of itself, a hypothesis echoed by others [2]. Conversely, human-written text is assumed to have a considerably more irregular log-probability function, where that function is taken over the perturbed passages of a given human-written passage.

Our essential approach, therefore, is to generate AI passages, perturb both human and AI candidate passages, then query the LLM we are trying to detect for its own log probability of having produced both the candidate and perturbed passages. If the log probabilities of the perturbed passages overall are consistently lower than that of the candidate, we classify it as AI-generated.

But unlike [1], which operated on GPT-2 and 3 and similar LLMs, we don't have access to ChatGPT's internal probability distributions, though OpenAI has released ChatGPT through its API. The novelty of our approach is to take the probability distributions from similar LLMs and compose them together to perform detection. In other words, we average the perturbation discrepancy generated by each of the query models.

The final discrepancy in the probabilities between the candidate and perturbed passages is thus measured with the function

$$\mathbf{d}(x, p_\theta, q) \triangleq \frac{1}{k}\sum_{i=1}^{k} \log p_{\theta_i}(x) - \mathbb{E}_{\tilde{x}\sim q(\cdot|x)}[\log p_{\theta_i}(\tilde{x})]$$

where p_{θ_i} is the probability output by query model i (where there are k total query models we are composing together), x and $\tilde{x}$ are the candidate and perturbed passages respectively, and q is the function which is used to perturb examples.

The code to create our dataset (tokenize the human-written text and use it to prompt ChatGPT for machine-generated text) was coded completely from scratch by us. We reference [1]'s original code for the perturbation and likelihood calculation portion of the project, but our code added documentation and heavily refactored the code for 1) readability and 2) the purpose of extending the code for our own project.

There are multiple natural baselines for our work: for one, we use the average of the query models' average token-wise log probability to determine if a candidate passage is machine-generated or not. Passages with high average log probability are likely to be generated by the model. The AUROC for this baseline for the current dataset we are working with is 0.83. The other baseline is to just query the one model most similar to ChatGPT whose probabilities we can access, GPT-3.5, whose results we report below.

2 Experiments

We utilize XSum [3], a dataset of news articles and one-sentence summaries. It's most often used for summarization training, but we utilize only the articles, since hard-to-detect AI-generated fake news and misinformation is a prime concern with AI-generated text. We prompted ChatGPT with "Complete the following text: " followed by the first 30 tokens of the human-written text as a prompt.

We use AUROC to evaluate detection, which measures the probability that the classifier ranks a random ChatGPT passage as more likely to be AI-generated than a random human passage.

Each experiment proceeds in three steps:

1. Perturbation: We use T5-3B to generate n perturbed examples for k pairs of one human, one ChatGPT candidate passage. For now, the hyperparameters used for this are to perturb 15% of each passage in spans of 2 words each, as these were determined to be most effective by [1]. This takes by far the greatest amount of time: e.g., generating 25 perturbations for 500 examples, so 25,000 perturbations, takes around 12 hours on a remote GPU.
2. Querying probabilities: Each perturbed and original passage is passed through the query models so the model assesses the log probability of each token: we average these as the overall log probability of each passage. For the perturbed passages, we take both the mean and standard deviations of the distribution of the log probabilities of the passages.
3. Evaluation: using the probabilities obtained at step two, calculate the Receiver Operating Curve and the AUROC score from that.

For our initial experiments, we queried just GPT-3.5 and with very small numbers of candidate examples and perturbations as a baseline test, achieving the following AUROC results:

	5 candidates	10 candidates
10 perturbations	0.242	0.256
20 perturbations	0.274	0.278

Figure 1: AUROC results for initial experiments on XSum dataset querying GPT-3.5 for probabilities.

These results, though they may not be representative as we scale further up in number of candidates and perturbations in future experiments, are still somewhat surprisingly low, as they indicate classification worse than random chance. Further investigation is required to understand why, but one current hypothesis is that ChatGPT's finetuned nature (for chat) may imply that its log probability distribution, which is what detectGPT relies on, may be significantly different than other LLMs, even neighbors like GPT-3.5 which we queried on for these preliminary experiments.

3 Future work

We plan to compose more query models together (set of language models listed above), instead of just using one query model (GPT-3.5). We also plan to expand the number of candidate passages and perturbations per candidate passage. If time permits, we want to test our model on datasets of other writing genres (e.g. medical question-answering, creative writing prompts). Another stretch goal is to tune the weights our model should give to the discrepancy returned by each query model by doing a grid search on a dev set of passages, instead of just averaging all discrepancies with equal weight.

References

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

[2] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2019.

[3] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.