DetectChatGPT: Black-Box Zero-Shot Detection of LLM-Generated 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`

Abstract

The detection of large language model (LLM) generated text has become an increasingly important problem, since popular LLMs like OpenAI's ChatGPT are producing human-like text that is difficult to distinguish from natural language, even to human evaluators. Recent research (Solaiman et al. (2019) Sebastian Gehrmann (2019) Mitchell et al. (2023)) have developed zero-shot methods for identifying LLM-generated text by using these models' internal probability distributions. However, many models, like ChatGPT do not release their internal probability distributions to the public. We adapt Mitchell et al. (2023)'s work on DetectGPT, the current state-of-the-art in zero-shot detection, by identifying ChatGPT-generated text even without access to the ChatGPT's internal probability distributions.
Our approach, tested on fiction and nonfiction passages written by humans and generated by ChatGPT, extends DetectGPT by querying LLMs similar to ChatGPT for probabilities and composing together their results. Surprisingly, we find that out of all possible combinations of different models, the most effective results come from using GPT-2 by itself as a query model, notably achieving as high as 0.99 AUROC on the WritingPrompts dataset, matching the benchmark of GPT-2 detecting its *own* passages in the original implementation of DetectGPT.

1 Collaborators

- Mentor: Lisa Li
- External Collaborators: Eric Mitchell
- Sharing project: N/A

2 Introduction

Large language models (LLMs) have become incredibly effective at producing human-like text in recent years, and human evaluation is no longer good enough to differentiate between AI-generated and human-written passages (Sebastian Gehrmann, 2019). Yet the reliable detection of AI-generated text remains a necessity in order to identify AI-generated student work, accurately assess learning, and prevent the proliferation of inaccurate news articles (Solaiman et al., 2019). As a result, recent research have focused on building models that, when given a candidate passage, will classify it as either machine-generated or human.

Most early work in AI-generated text detection attempted to train a classifier using classical supervised learning methods such as logistic regression, random forest, and SVM. While such methods can be quite effective with classifying text in certain domains Bakhtin et al. (2019), a key issue with such methods, as raised by Bakhtin et al. (2019) Uchendu et al. (2023), is that unless trained on an enormous and diverse corpus of text similar to an LLM training dataset, the classifiers often overfit to the domains of text they are trained on.

For this reason (and our resource limitations), we pursue zero-shot detection methods which don't require training. The current state-of-the-art in zero-shot AI-generated text detection is Mitchell et al. (2023)'s DetectGPT. DetectGPT operates largely in a 'white-box' setting, with internal access to the log probabilities per token of the *detected model*, i.e. the LLM whose text we are trying to detect. DetectGPT performs well on fairly recent LLMs such as GPT-3, but its authors noted that one of its weaknesses is that its functionality depends on being able to query an LLM for these log probabilities.

Unfortunately, not all widely-used LLMs release their internal probability distributions. For example, although OpenAI has recently released ChatGPT through its API, ChatGPT's log probabilities are still not accessible as of this report's writing. Detecting ChatGPT text, therefore, requires adapting DetectGPT to a 'black-box' setting.

Our contribution is extending DetectGPT to this black-box setting to develop **detectChatGPT**, which will classify a given candidate passage as either human or ChatGPT-generated. Since we cannot query the ChatGPT's internal distribution, we propose querying the internal distributions of other LLMs similar to ChatGPT (such as OpenAI's other GPT models), and composing the results from these other LLMs to approximate the internal distribution of ChatGPT. In other words, in our detectChatGPT model, the *detected model* (ChatGPT) is different from the *query model(s)*, whose internals we access for detection.

Interestingly, the best version of DetectChatGPT is using results from only GPT-2, rather than composing the results from multiple different LLMs. Nonetheless, the best version of DetectChatGPT achieved great performances on par with the original DetectGPT across multiple different datasets and ChatGPT hyperparamters.

3 Related Work

There are, broadly speaking, two approaches to detecting LLM-generated text. One approach is to use supervised learning to train a classifier that can discriminate between AI-generated and human text. Most classifiers use a bag-of-words text representation and train traditional classifiers such as logistic regression, random forest, and SVM (Solaiman et al., 2019; Fagni et al., 2021). Bakhtin et al. (2019) adapted and trained an energy-based neural model originally developed for detecting computer-generated images to detect computer-generated text.

While these methods can achieve good performance, they suffer from two main weaknesses. First, training classifiers requires curating a dataset of machine-generated and manually-written texts, which can be expensive and time-consuming. Second, the content of the dataset has a great deal of influence over the resulting model. Bakhtin et al. (2019) found that models trained specifically to detect machine-generated text have a tendency to overfit to the subject domain of the training dataset, and do not generalize well across different text-generation models.

Thus the alternative approach pursued in more recent strains of work is performing detection by using zero-shot detection methods. These methods exploit certain features of how LLMs produce natural language to perform detection without training. Solaiman et al. (2019) applied a simple threshold on the total probability of the passage according to the source model's probabilities, which outperformed simple trained classifiers with an accuracy between 83 and 85% and provided a baseline for future zero-shot detection methods. Sebastian Gehrmann (2019) tested a number of possible zero-shot features, such as the probability of a randomly sampled word given the previous words and the absolute rank of a word, with the highest AUROC score being .87.

While these zero-shot methods rely on sampled or averaged probabilities of single tokens within the text, Mitchell et al. (2023)'s DetectGPT devised a new zero-shot method to consider the local structure of the probability function around a given text. 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 Holtzman et al. (2019). 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. This new zero-shot method

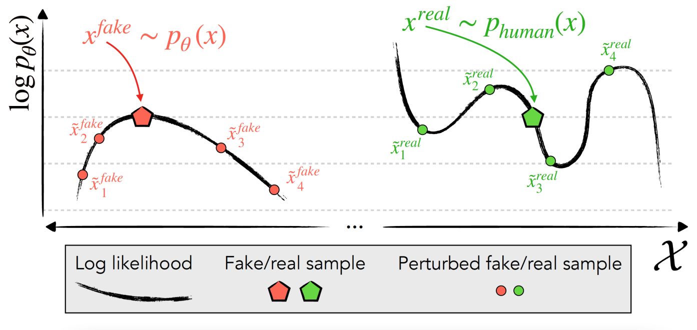


Figure 1: This figure, taken from Mitchell et al. (2023), visually conveys the hypothesis behind DetectGPT. For an AI-generated passage and its perturbations (in red, on the left) we hypothesize that the candidate passage (pentagon) always has a higher probability compared to its perturbations (dots), according to the LLMs queried. We hypothesize that this is *not* the case for human passages (in green, on the right), for which we assume perturbations may have higher or lower probabilities.

outperformed existing methods with .95 AUROC, and is considered current SOTA for zero-shot detection.

4 Approach

Our approach extends Mitchell et al. (2023)'s DetectGPT. To detect AI-generated texts, DetectGPT collected human and AI-generated candidate passages, generated perturbed versions of these candidate passages, and then queried for the overall log probability of the candidate and perturbed passages from the LLM suspected of generating the candidate passage. 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 DetectGPT, which operated on GPT-2, GPT-3 and similar LLMs to detect their own text, we don't have access to ChatGPT's internal probability distributions. 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 together the perturbation discrepancy scores from multiple *query models* in order to estimate the probability distribution from the *detected model*.

We describe our procedure below in further detail. Each experiment proceeds in five steps.

1. **Generating ChatGPT passages:** For each passage in a given dataset of human-written passages, we generate its ChatGPT counterpart by using the first few tokens of the human-written passage.

2. **Perturbation:** We use a language completion model to generate n perturbed examples for each human-written and ChatGPT-generated passage in our dataset. By using a model to generate perturbations, we minimize changes to the passage's meaning. An example of a perturbation would be:
   Candidate **"Pizzolato, who has dual Brazilian and Italian citizenship"** $\rightarrow$
   Perturbed **"Pizzolato, who holds both Brazilian and Italian citizenship"**

3. **Querying probabilities:** Each perturbed and original candidate passage is passed through the query models we have selected. Each model assesses the log probability of each token in the passage, and we average these to find the overall log probability of each passage. For each query model, we also record the mean and standard deviation of the distribution of overall log probabilities of the n perturbed passages.

4. **Calculating discrepancy:** The final discrepancy in the probabilities between the candidate and perturbed passages is thus measured with the function

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

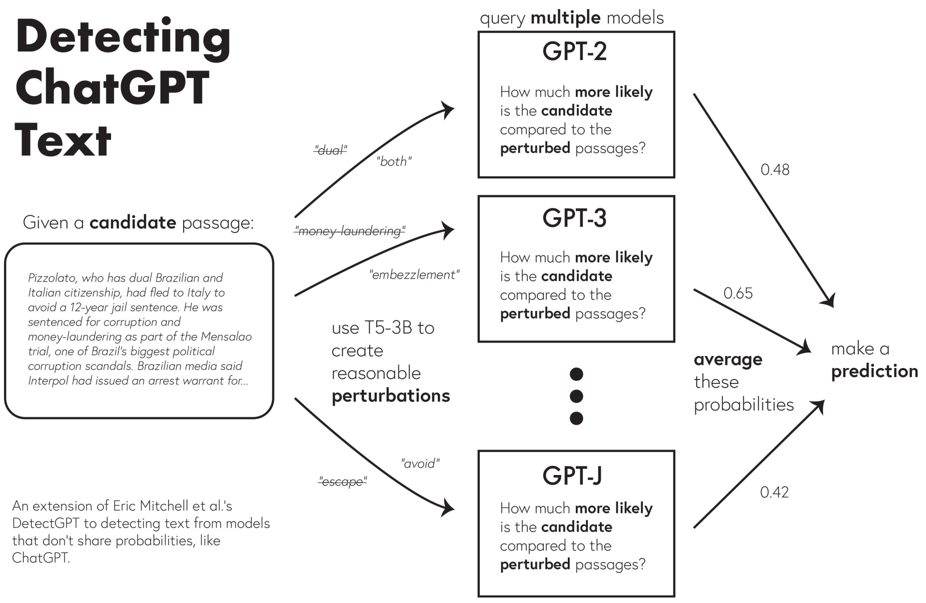

Detecting ChatGPT Text
query multiple models
GPT-2
How much more likely is the candidate compared to the perturbed passages?
GPT-3
How much more likely is the candidate compared to the perturbed passages?
GPT-J
How much more likely is the candidate compared to the perturbed passages?
Given a candidate passage:
Pizzolato, who has dual Brazilian and Italian citizenship, had fled to Italy to avoid a 12-year jail sentence. He was sentenced for corruption and money-laundering as part of the Mensalao trial, one of Brazil's biggest political corruption scandals. Brazilian media said Interpol had issued an arrest warrant for...
~~"dual"~~ "both"
~~"money laundering"~~ "embezzlement"
~~"escape"~~ "avoid"
use T5-3B to create reasonable perturbations
0.48
0.65
0.42
average these probabilities
make a prediction
An extension of Eric Mitchell et al.'s DetectGPT to detecting text from models that don't share probabilities, like ChatGPT.

Figure 2: The overall pipeline for one candidate passage through our classifier.

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.

5. **Evaluation:** The predictions from the first four steps output probabilities between 0 and 1. Since we use AUROC for evaluation, we calculate the receiver operating curve over numerous classification thresholds ϵ and calculate the area.

The code to create and process our data was coded completely from scratch by us. We reference Mitchell et al. (2023)'s original code for the perturbation and likelihood calculation portion of the project (which can be found here), but our code added documentation, heavily refactored code for readability, and added several new features to extend the code for our own project. Our own repository can be found here.

5 Experiments

5.1 Data

We used two different datasets of human-written passages to represent both nonfiction writing and fiction writing. We included datasets from both nonfiction and fiction texts since a commonly cited advantage of the zero-shot method its its generalizability across different writing domains, and we wanted to test that our new zero-shot method maintained this strength.

The nonfiction dataset was XSum (Narayan et al., 2018), a dataset of news articles and one-sentence summaries. It's most often used for summarization training, but we utilized only the articles, since hard-to-detect AI-generated fake news and misinformation is a prime concern with AI-generated text. The fiction dataset was the Reddit WritingPrompts (Fan et al., 2018), a dataset of fictional passages written in response to a prompt, scraped from the site forum Reddit. From each of these datasets, we randomly selected 500 passages (as did the original DetectGPT paper) to run our experiments on. Selecting a small subset was necessary as our method (generating ChatGPT counterparts, querying for likelihoods from multiple query models, etc.) is quite expensive compared to our limited compute resources. For each passage in these smaller datasets, we generated the passage's ChatGPT counterpart.

5.2 Evaluation

As DetectChatGPT is fundamentally a binary classifier, we use AUROC scores to evaluate its performance. Other AI-generated text detection methods in the literature commonly use an AUROC score to report their performance as well (Mitchell et al., 2023; Sebastian Gehrmann, 2019; Solaiman et al., 2019). AUROC measures how likely the model, given a random AI-generated and a random human passage, will predict that the AI-generated passage is more likely to be fabricated than the human passage.

Since, as far as we know, black-box zero-shot detection like ours approach has not been tried in the literature, we use two natural baselines. For one, we compare to Mitchell et al. (2023)'s results on the same XSum and WritingPrompts datasets in the white-box setting; which achieved as high as 0.99 AUROC for both WritingPrompts and XSum datasets; this high result occurred when querying GPT-2 to detect GPT-2-generated passages. Two, we conduct baseline experiments that use just *one* query model for detection, instead of composing models together.

5.3 Experimental details

Generating ChatGPT passages: For each passage in our two human-written datasets (XSum and WritingPrompts), we generated the passage's ChatGPT counterpart in the way that felt most natural to the dataset. For XSum's news articles, there was no natural notion of a prompt, so we just prompted ChatGPT with "Complete the following text: " followed by the first 30 tokens of the original human-written passage as a prompt, following Mitchell et al. (2023). For WritingPrompts, since the dataset already contains a natural notion of prompt, we prompted ChatGPT with, "Write a short story based on the following prompt: {prompt}", inserting the actual prompt in the brackets.

OpenAI's ChatGPT API also offers control over a "temperature" hyperparameter, which controls how randomly ChatGPT samples from its own probability distribution over tokens while producing tokens. Because perturbing passages and querying probabilities is so expensive, we chose to use temperature as the key hyperparameter for our experiments and keeping others fixed in place from Mitchell et al. (2023), hypothesizing that higher temperature texts might have probability distributions that look more "human-like" (in the sense of Figure 1), and therefore would be harder to detect with our method.

For each passage, we generated its ChatGPT counterpart with a temperature of 0, 0.5, and 1: though temperature goes as high as 2, we found that examples with temperature above 1 were essentially gibberish, and easily detectable by humans. This results in a total of 6 experiments with 6 different datasets, XSum and WritingPrompts each taken at three temperatures.

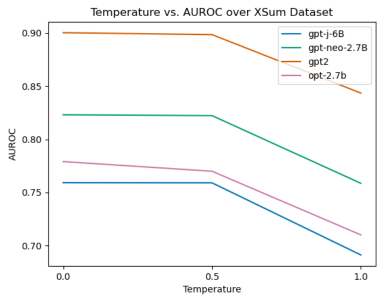
Perturbation: The language completion model we chose to generate our perturbations was T5-3B, which was used in DetectGPT. For each passage, we generated 100 perturbed versions of that passage. The hyperparameters for perturbation (how many words to perturb at a time, and how much of the text to perturb overall) are also important in testing the resiliency and effectiveness of this detection model, but due to our resource limitations and the fact that Mitchell et al. (2023) effectively swept across many hyperparameters in their experiments, we followed their hyperparameter recommendations: we perturbed 15% of each passage in spans of 2 words each, doing 100 perturbations for each candidate. This took by far the greatest amount of time: generating 100 perturbations for 1,000 candidate passages, i.e. 100,000 perturbations, takes around 12 hours on a remote GPU.

Querying probabilities: For our experiments, the query models we chose were GPT-Neo, OPT-2.7B, GPT-J-6B, GPT-2, and GPT-3. These models were chosen because they were identified as being similar to other GPT models (Mitchell et al., 2023) and because their internal probability distributions were available for querying. Due to resource limitations, GPT-3 (the 'babbage' variant in the OpenAI API) was only queried for two of the six datasets, XSum and WritingPrompts at temperature 0.

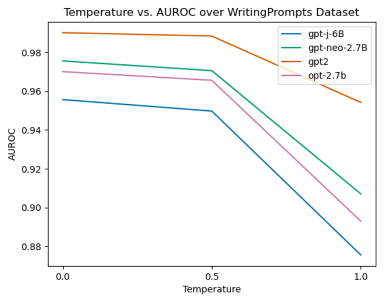
Calculating discrepancy: Once we queried likelihoods from all query models, we tested all possible combinations of these 5 query models, for $k = 1$ to $k = 5$ (where k is the number of query models we are composing together). In our results table below, we report only the results for $k = 1$ and the results of the best performing combination out of every single query-model-combination that we tested.

Table 1: AUROC scores for detecting ChatGPT-text across XSum and WritingPrompts datasets, with ChatGPT passages generated at temperatures 0.0, 0.5, and 1.0. Due to resource limitations, GPT-3 was only queried for two out of the six datasets.

	XSum			**WritingPrompts**		
Query Model	0.0	0.5	1.0	0.0	0.5	1.0
GPT-Neo	0.823	0.822	0.759	0.976	0.971	0.907
OPT-2.7B	0.779	0.770	0.710	0.970	0.966	0.893
GPT-J-6B	0.759	0.759	0.691	0.956	0.950	0.875
GPT-2	**0.900**	**0.898**	**0.843**	**0.990**	**0.989**	**0.954**
GPT-3	0.798	–	–	0.965	–	–
Best Comp.	0.900 **(GPT-2)**	0.898 **(GPT-2)**	0.843 **(GPT-2)**	0.990 **(GPT-2)**	0.989 **(GPT-2)**	0.954 **(GPT-2)**



(a) AUROC plotted against temperature on XSum dataset.



(b) AUROC plotted against temperature on WritingPrompts dataset.

Figure 3: AUROC vs. temperature on both WritingPrompts and XSum datasets for all models.

5.4 Results

Table 1 showcases the AUROC results for our baseline black-box detection experiments on a single query model, and the 'Best Composition' row gives the best result over all single query models and all possible combinations of all the query models.

Surprisingly, one of our baselines outperformed all of the others and all of the experiments on combinations of models. GPT-2 on its own outperforms all other query models and query model combinations at text detection across temperatures. In the case of the WritingPrompts dataset at temperature 0 (produced essentially deterministically by ChatGPT), GPT-2 actually matches the aforementioned DetectGPT baseline of detecting its *own* text, with an AUROC of 0.99. Though we did not have the resources to query GPT-3 for all six of our datasets, even for those datasets where we did query GPT-3, GPT-2 *still* outperformed all other models. As the candidates from ChatGPT get more random at higher temperatures, performance does degrade somewhat (as can be seen in Table 1 and Figure 3) for all models, confirming our hypothesis; that said, detection does not at all come close to breaking down, still achieving a 0.843 and 0.954 AUROC on XSum and WritingPrompts respectively at the highest temperature. (The full results for all combinations of models are available in our aforementioned repository.)

6 Analysis

The result that most immediately demands explication is, of course, the shocking effectiveness of GPT-2 at text detection, all the more surprising given that it is thousands of times smaller than

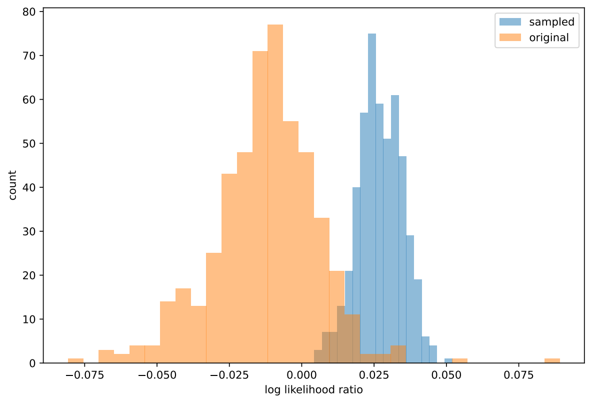


Figure 4: For the WritingPrompts dataset at temperature 0, the distributions of log-likelihood ratios between the likelihood of the candidate and the average likelihood of the perturbed passages, for ChatGPT-sampled and original passages.

ChatGPT/GPT-3.5, with parameters in the millions instead of the billions. In Figure 4 we observe that using GPT-2 for probability querying on the WritingPrompts dataset at temperature 0, for which we obtained 0.99 AUROC, results in a probability-ratio distribution that exactly matches the original hypothesis of DetectGPT. In blue, we observe that the log likelihood ratios of the probability of the candidate passage to the probability of the perturbed passages are always positive: that is, over all the ChatGPT candidates, the candidate passage's likelihood is always greater than the average likelihood of the perturbed passages, mirroring the hypothesis of Figure 1. On the other hand for the human passages, some of the likelihood ratios are negative and some positive, with the distribution over candidates centered just under a log ratio of 0, i.e., a ratio of 1. So in this case, detection of ChatGPT text works with GPT-2 as a query model in the exact same way white-box detection worked in the original DetectGPT.

That said, it does not appear that GPT-2 is simply simulating the probabilities that were given by ChatGPT. Incredibly, when looking at the raw log-likelihoods of the candidate passages by themselves in Figure, it seems that GPT-2's likelihood of the ChatGPT perturbations **over all candidates** has basically the exact same distribution as its likelihood assessment of the ChatGPT candidates themselves. This is quite different from what would be expected if we could query ChatGPT itself: its likelihood distribution over the candidates would in general be greater than that of the perturbations, especially at the graphed temperature (0), because it produced those passages itself. GPT-2's success therefore cannot be explained because it is somehow 'simulating' ChatGPT and is outputting the same probabilities as ChatGPT would. This makes sense considering that GPT-2 is arguably the least similar model to ChatGPT, at least in terms of sheer size, out of all the query models we used.

It is all the more remarkable, then, that GPT-2 successfully outputs the perturbation discrepancy visible in Figure 4. This suggests that the detection method pioneered by Mitchell et al. (2023) does not in itself depend on the query model being able to 'recognize' its own passages, i.e., give the passages it produces a high likelihood. That said, comparing the human and ChatGPT examples, it does 'recognize' the ChatGPT passages (candidates and perturbations) as overall being more likely than the human ones.

The perturbation discrepancy that GPT-2 produces may in fact depend on other confounding factors, which demand further interpretability work. In general, in both the original DetectGPT and our work, we observe that detection works better on WritingPrompts than XSum. One might hypothesize that may be because the overall discrepancy between human and ChatGPT candidates for WritingPrompts is simply much greater: since it's prompted with a creative prompt ChatGPT can go in a much different direction than the human counterpart compared to when it's asked to complete an existing piece of news text. (ChatGPT might be especially provoked into creating a human-like response for XSum data if the prompt contains specific proper nouns.) Therefore, one might hypothesize that in

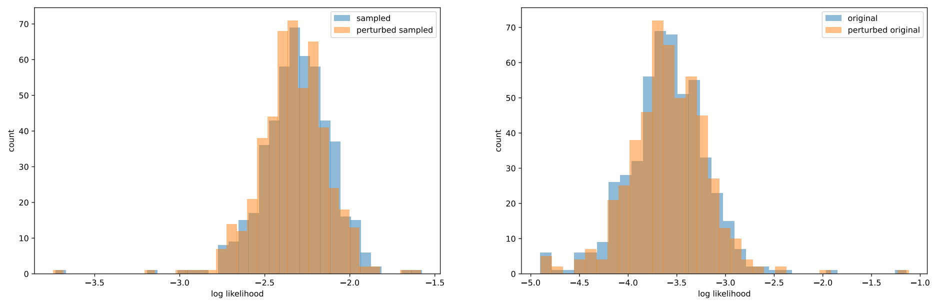


Figure 5: Raw log likelihoods of candidates (blue) and perturbations (orange) for both ChatGPT (left) and human (right) examples, for GPT-2 querying on the WritingPrompts dataset temperature 0.

general there's a bigger discrepancy in how likely GPT-2 thinks human WritingPrompts vs. ChatGPT WritingPrompts candidates are. This overall discrepancy is not necessarily backed up by the data, however: for WritingPrompts and XSum at temperature 0, for example, the average difference in likelihoods that GPT-2 outputs for a ChatGPT passage vs a human passage is about 1.1 in favor of ChatGPT for both datasets.

A more reasonable hypothesis, though not testable with the data we have currently available, is the fact that since our perturbations are of length 2, in practice the perturbations simply take the form of swapping out one word for a synonym instead of all-out rephrasings of a sentence or paragraph. It it possible that one reason that GPT-2 detects ChatGPT's perturbation discrepancy so well is that GPT-2's internal token-vocabulary model is the most similar to ChatGPT's. That is, both GPT-2 and ChatGPT agree that a word is much more likely to occur than its synonyms over a much broader range of words. Because of resource limitations we could not perturb at different lengths, but if perturbing at longer lengths worsened GPT-2's performance, it could be the beginnings an indication that GPT-2's success depends on it having the same individual word *preferences* as ChatGPT and that in general, for any query and detected model, this method may work if these word preferences similarly line up.

7 Conclusion

While our work's original plan was to compose multiple different LLM's results to approximate the results we would have gotten from querying ChatGPT's internal probability distribution, we found that the best performing version of DetectChatGPT was one that only relied on GPT-2 to approximate ChatGPT. This version of DetectChatGPT performed well with an AUROC score of .9 for nonfiction writing from the XSum dataset and .99 for fictional writing from the WritingPrompts dataset. This performance is nearly similar to (and in WritingPrompt's case, is on par with) Mitchell et al. (2023)'s DetectGPT's performance, even though DetectChatGPT was expected to perform lower as it works in a black-box setting (as opposed to a white-box setting). To that regard, we have succeeded in developing a method that can detect when a passage has been generated by a LLM that does not release its internal probabilities.

One limitation is that our model, while it requires less resources than a classifier trained with supervised learning in that it doesn't need a training dataset, is incredibly compute intensive. Generating perturbations and querying for the likelihood of all perturbed passages from several different models required a significantly higher amount of storage, compute power, and money than expected.

The other limitation follows from the first model: we have only been able to test our results on ChatGPT specifically due to resource constraints, and so we are uncertain if the precise results of this work extend to LLMs other than ChatGPT. While the viability of using other LLMs to estimate an unknown LLM has been shown, it may not be the case that solely using GPT-2 specifically will be the best way to approximate LLMs other than ChatGPT that emerge in the future. We hope to see similar experiments that test the detection of other LLMs and see whether our exact results are generalizable to LLMs other than ChatGPT.

References

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

Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. 2021. Tweepfake: About detecting deepfake tweets. *Plos One*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *ACL Anthology*.

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

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. *ArXiv*, abs/1808.08745.

Alexander M. Rush Sebastian Gehrmann, Hendrik Strobelt. 2019. Gltr: Statistical detection and visualization of generated text. *ACL Anthology*.

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. *ArXiv*.

Adaku Uchendu, Thai Le, and Dongwon Lee. 2023. Attribution and obfuscation of neural text authorship: A data mining perspective.