**Detecting LLM-generated Text with a Trinary Classifier Based on Sentence Completions**

**Abstract**

Detecting text generated by large language models (LLMs) is a problem that is only becoming more important and difficult as LLMs improve and become more common. Many solutions of different types have been proposed for this problem. This paper proposes an LLM-generated text detector that takes sentences from an input text, removes the ends of the sentences, and then asks ChatGPT, Bing Copilot, and Google Gemini to complete the sentences. Similarity between the original sentence and the generated sentences is used to classify the input into one of three categories: High confidence LLM-written, Medium confidence LLM-written, and Human-written. Human written texts and LLM-generated texts are given to this LLM-generated text detector. The results it produces are then evaluated, with particular attention paid to which texts ended up in the Medium confidence LLM-written category, and further improvements are discussed.

**Introduction**

Determining whether a piece of writing is written by a human or generated by a large language model (LLM) is becoming increasingly important in the current era as LLMs become more popular and more potent. One of the most important reasons for doing this is the increase in use of LLM-generated text to mislead people. According to Hanley and Durumeric [1], there was a 474% increase in LLM-generated articles on misinformation sites between January 1, 2022 and May 1, 2023. Over the same period, there was even a 57.3% increase in LLM-generated articles on mainstream news websites (though this increase was mainly from smaller websites). Wu et al. [2] mention another problem arising from LLM-generated text: recursive degradation. This happens because earlier LLM-generated text gets used as training data for newer generative AI models, resulting in the newer models having lower quality output. Additionally, Wu et al. [2] mention several other issues that LLM-generated text causes. One of these is the use of LLMs in academic writing, which can make it more difficult to evaluate students in educational settings and is harmful to academic integrity. Another issue that Wu et al. [2] covers is the possibility for LLM-generated text to reduce the richness of human communication. As a whole, there are many reasons why determining whether text was written by a human or an LLM is important.

This report will focus on three LLMs: ChatGPT, Bing Copilot, and Google Gemini. These three LLMs were picked because ChatGPT and Copilot are associated with Microsoft and Gemini is associated with Alphabet (Google’s parent company), and Microsoft and Alphabet were considered by Rudolph et al. [3] to be the leaders of the “War of the chatbots”.

Solutions to the LLM-generated text detection problem traditionally involve binary classification; a text is classified as either human-written or LLM-written [2]. The proposed solution described in this report involves a trinary classification system where the three levels are High Confidence LLM-written, Medium Confidence LLM-written, and Human-written. Input sentences will be classified into one of these categories based on taking these sentences, removing the second half of the sentences, asking the three LLMs mentioned above to complete the sentences, and comparing the results to the original texts. The idea for dividing the LLM-written classification into two levels is to allow further steps to be taken if the text is classified as Medium Confidence LLM-written. These further steps can vary depending on the application. This is done because false positives in binary classification systems (“positive” referring to classifying a text as LLM-written in this scenario) can have significant consequences for the human writers involved in situations such as written assignments in educational settings [4]. The trinary classification system allows for more flexibility for dealing with potential false positives.

**Background**

Many different solutions have been proposed for solving the LLM-generated text detection problem. Wu et al. [2] divide these solutions into distinct categories. One of these categories is watermarking techniques. Watermarking techniques involve embedding patterns in text and later detecting these patterns, but they rely on access to the deployment of the LLM in question.

A second category described by Wu et al. [2] for LLM-generated text detection is statistics-based methods. These methods use data from the text they receive to identify patterns, and then analyze the data and patterns to determine whether or not the text was written by an LLM. One advantage of using statistics-based methods is that they do not require specialized access to a given LLM. Wu et al. [2] further subdivide statistics-based methods into Linguistic Feature Statistics, White-box Statistics, and Black-box Statistics. Linguistic Feature Statistics, as the name suggests, involves taking linguistic features from the text, such as usage of phrases or frequencies of n-grams, to detect LLM generation. White-box Statistics methods, which can only be implemented if there is direct access to the text’s source LLM, measure the characteristics of a text related to the logits that LLMs produce to determine whether or not the text is LLM-generated. Two common baseline detection methods, Log-likelihood, and Rank, are White-box methods. Black-box Statistics methods do not access any logits, but are still able to score texts based on various features to detect LLM generation. The project described in this paper is an example of a Black-box method.

The third category of LLM-generation detector, according to Wu et al. [2], is neural-based methods. These methods are subdivided into Feature-Based Classifiers, Pre-training Classifiers, and LLMs as Detectors. Feature-Based Classifiers are models that can be trained on linguistic features such as word frequencies or sentence structures, or LLM features that, similar to White-box Statistics methods, require access to LLM logits. Pre-training classifiers include encoder-based classifiers, contrastive learning, adversarial learning methods, and features-enhanced approaches. Encoder-based classifiers, which are fine-tuned to detect LLM writing, are particularly notable because they include RoBERTa, which Wu et al. [2] refer to as “a robust baseline” among LLM-generated text detectors. Additionally, SimLLM from Nguyen-Son et al. [5], which takes individual sentences as input, uses LLMs to proofread them, and then compares the proofread sentences to the originals, is considered an encoder-based classifier [2]. Other methods use LLMs themselves as detectors of LLM-generated text. Though using the LLMs directly to detect text that they themselves wrote is unreliable, Instructional Contextual Learning has been shown to be more reliable. In Instructional Contextual Learning, examples are included in prompts provided to an LLM in order to help the LLM learn.

**Related Work**

As mentioned in the background section, the method discussed in this paper is a Black-box Statistics method. Several previous methods used to solve the LLM-generated text detection problem can be considered Black-box Statistics methods. One from Yang et al. [6] is called DNA-GPT, and it cuts input texts in half, sends the first half as input to LLMs in order to regenerate the second half, and compares the new texts with the old texts using N-gram analysis. This method performed well, outperforming OpenAI’s classifier on five different datasets. Another Black-box method by Mao et al. [7] called Raidar was based on the tendency of LLMs to prefer LLM-generated text over human-written text in terms of quality. When asked to rewrite input text, the LLMs were found to make more changes with human-written text than with LLM-generated text. Raidar was able to achieve high F1 detection scores. A Black-box method from Wu et al. [8] named GECScore used the fact that text written by humans tends to have more errors in grammar than LLM-generated text to detect LLM-generated text. This method had a 98.7% average AUROC.

**Problem Definition**

Formally, in this paper, an LLM-generated text is defined as text that was produced using only a large language model. None of the words in the text came directly from the mind of a human. The only human input for LLM-generated text would be writing a prompt to the LLM that results in the generation of the text. This means that text that was created through a mix of LLM generation and human input (such as an essay that was initially written by a human, but later revised using an LLM) is not covered by the problem in this paper.

When evaluating texts, similarity between an original text and new text from an LLM is defined using cosine similarity. Cosine similarity is a popular method for comparing similarity between texts.

There are several technical challenges involved in my solution to the LLM-generated text detection problem. One big challenge is determining cutoffs for the trinary classification system for texts that will actually make the classification system useful. For example, what is a good numerical cutoff for determining whether a similarity score is “High Confidence LLM-written” or “Medium Confidence LLM-written”? I believe that resolving this will require trial and error and multiple iterations of determining and reevaluating the cutoff values for the three classification levels until these values are ideal.

Another technical challenge for the solution will be automating as much of it as possible. The solution could be done manually, but that would be very slow and impractical with large datasets. Automating the solution will make it much faster, but that will require using the APIs of the LLMs. The problem is that I have no experience interacting with LLM APIs, and this project will require interacting with three of these APIs. Due to my lack of experience and the potential financial cost of dealing with the APIs, there is a chance I will not be able to automate my entire solution. However, even automating part of it will be a significant improvement over doing all of it manually.

**Example**

The following example gives a general idea of what I plan to do for this project. This is an example of a sentence written by ChatGPT:

“Northern cardinals are known for their beautiful, melodic songs, which both males and females sing.”

I take this sentence and remove the ending portion of it. For now, I will remove the second half of the sentence. The halves are determined by the word count, and the example sentence has 15 words, so I keep the first 7 words (I chose to round down when splitting the count in half) and give these 7 words to ChatGPT, prompting it to complete the sentence.

This is the prompt I gave to ChatGPT:

“Please complete the following sentence: "Northern cardinals are known for their beautiful,"”

This is the completed sentence ChatGPT gave to me in response:

“Northern cardinals are known for their beautiful, vibrant red plumage, melodious songs, and striking crests.”

Now that I have a new completed sentence, I compare it to the original sentence. The cosine similarity of the two sentences is 0.5853694070049635. I then repeat the process, keeping the first 8 words of the original sentence instead of only the first 7 words. This time, when asked to complete the sentence, ChatGPT produced the following:

"Northern cardinals are known for their beautiful, melodic songs that consist of a variety of whistles, trills, and clear, sharp notes."

The cosine similarity between this and the original sentence is 0.6227991553292184. I repeat the process again, this time keeping the first 9 words. ChatGPT’s completed sentence is:

“Northern cardinals are known for their beautiful, melodic songs, which they use to communicate with mates, establish territory, and signal alarms to other birds."

The cosine similarity between this and the original sentence is 0.6030226891555273. For now, I plan to repeat this process until the number of words kept is the two-thirds the length of the original sentence (so the first time the prompt’s length is two-thirds or more of the original sentence length is the last prompt length used). Another thing I need to plan for is the effect the amount of words kept might have on the cosine similarity. With more words kept, the cosine similarity might tend to increase, but this is only because more words at the beginnings of the two sentences are the same.

For each sentence, I will get the cosine similarities for ChatGPT, Copilot, and Gemini. I will save the highest cosine similarity for each of the three LLMs and average these three similarity scores. This will be used to classify the sentence into one of three categories: High Confidence LLM-written, Medium Confidence LLM-written, or Human-written. The High Confidence LLM-written classification will be given priority. If the cosine similarity lands on the boundary between Medium Confidence LLM-written and High Confidence LLM-written, the sentence will be classified as High Confidence LLM-written. Similarly, Medium Confidence LLM-written will take priority over Human-written. I will repeat the process with the next sentence, and continue to repeat this process for the rest of the sentences.

**Approach**

The approach has already been described in previous sections, but I will put it all together here. Basically, I am taking sentences, some generated by a human and some generated by an LLM, and classifying them into one of three categories: High Confidence LLM-written, Medium Confidence LLM-written, or Human-written. The metric I am using for this classifier is cosine similarity between the original sentence and sentences generated by putting the first portion of the original sentence in an LLM and asking it to complete the sentence. The first portion varies from the first half, based on word count, of the original sentence, rounded down, to the first time the word count is two-thirds or more of the original sentence, inclusive. For example, if the word count in the original sentence is 13, then I will use the following prompts: first 6 words, first 7 words, first 8 words, and first 9 words. In this process, I am assuming that higher cosine similarity scores mean there is a higher chance that the sentence was LLM-written. I will use three different LLMs for this: ChatGPT, Copilot, and Gemini. I will take the highest cosine similarity score associated with each LLM and average these three values. This is the value that will ultimately be used in the classifier. My goal is to minimize the number of false positives (human-written sentences classified as high confidence LLM-written) while still rewarding true positives and true negatives and punishing false negatives.

One limitation of this approach is that using cosine similarity only might not be the best metric for this task. Using cosine similarity alongside other metrics might have been better. Another limitation is that LLMs have randomness associated with them. High cosine similarity scores and low cosine similarity scores might only happen because of luck.

**Implementation**

The preceding Approach section gives a general idea of my process, and here in this section are some additional details about the process. The code for this experiment was written entirely in Python. I used the nltk and math libraries, and also had to download punkt and stopwords from nltk. Additionally, I imported stopwords from nltk.corpus and word\_tokenize from nltk.tokenize. I intended to automate prompt generation for the three LLMs, but due to time and cost ultimately decided not to. Instead, I manually entered prompts for the three LLMs. The prompt I used always had the same format: Please complete the following sentence: “first\_part\_of\_sentence”.

I ended up using a total of 22 sentences: 11 generated by ChatGPT, and 11 written by me for various past assignments in past courses. I chose to only include sentences written by me because sentences from well-known works might already be known to the LLMs and I didn’t want to use something I did not have permission to use. I wanted to also include 10 sentences generated by Copilot and 10 sentences generated by Gemini, but I didn’t have the time to go through all of the steps in my process manually for 20 additional sentences.

The versions used for ChatGPT were 4o and 4o mini (more on why there were 2 different versions in the Discussion section below), the version of Copilot used was Quick response, and the version of Gemini used was 2.0 Flash.

**Evaluation**

The final boundaries I came up with for my classifier were as follows:

With these ranges, I had 0 false positives (human-written sentences classified as High confidence LLM-written) and 2 false negatives (LLM-generated sentences classified as Human-written). There were 2 true positives (LLM-generated sentences classified as High confidence LLM-written) and 9 true negatives (human-written sentences classified as Human-written). I did not count anything ending up in the Medium confidence LLM-written as a false positive or false negative. The other 9 sentences, 2 of which were human-written and 7 of which were LLM-written, were placed in the Medium confidence LLM-written category. While the true positive and false negative counts were not good, I consider the 0 false positives as a good result. My intention was to prioritize lowering false positives as much as possible. Human-written text ending up in the Medium confidence LLM-written category just means that further steps should be taken to ultimately determine the source. Going from 22 sentences down to 9 to take further steps on meant that over half of the sentences do not need any further action; this means, at the very least, my classifier can be used as a filter. While 2 LLM-written sentences did pass through the filter as human-written, I consider even 1 false positive as worse than numerous false negatives. I tried to balance out making something that actually has use and does not just automatically declare every sentence it receives as a negative (human-written) while still keeping the false positives to a minimum.

**Discussion**

One limitation of my implementation was the small amount of data used: only 22 sentences total. This was mainly due to not automating the process and the time it took to manually do everything; therefore, an obvious future improvement is automating at least part of the process and using more data. Another limitation, as I mentioned in the Approach section, is relying only on cosine similarity when I could have used multiple different metrics instead. I also noticed that sometimes two sentences would have different word choices but the cosine similarity would still be 1.0. Another factor was the length of the original sentence kept in the prompt might have affected the cosine similarity scores, which I mentioned in the Example section.

Another possible limitation is the rewards and punishments I used for classifications were arbitrary. My focus with these values was minimizing the false positives, which I did manage to achieve, but it is possible that the rewards and punishments I chose led to the 2 false negatives.

Other limitations and simply points of interest have to do with my experience using the LLMs themselves. With ChatGPT, sometimes the site would say that I could not use version 4o anymore for a specified time with my free plan and would automatically downgrade to 4o mini, but then it would upgrade up back to 4o before the specified time had passed. So some responses were written with 4o and some with 4o mini. Another thing that was true of all three models at some point was using references from various websites in their answers. This could affect the results of the cosine similarity because the references were most likely written by humans and that could make the LLM’s output more human-like. Additionally, I noticed that sometimes the models would provide a paragraph or more in response to my prompts even though I just asked for a sentence. This was particularly notable for Copilot. Copilot would also occasionally change the initial part of the sentence and sometimes this happened even if I gave it the same prompt twice. This could have resulted in lower cosine similarity scores for my classifier because I was using the average of all three models’ highest. Also, even though the LLM-generated sentences I used were all from ChatGPT, on many occasions one of the other two models would have higher cosine similarity scores, typically Gemini. This might have been because it seemed like Gemini had a tendency to sometimes keep its answers shorter.

Another limitation was the origin of the source material. I only used ChatGPT for LLM-generated sentences, and only used text written by myself for the human-written sentences. The results might be different with more variance in the sources.

**Conclusion**

Determining whether text is LLM-generated or human-written is becoming an increasingly important issue. There are many binary classifiers already out there, but I created a trinary classifier that would classify text into one of three categories: High Confidence LLM-written, Medium Confidence LLM-written or Human-written. I tested this on 22 total sentences: 11 from ChatGPT and 11 from my own writing. In the end, 0 of the human-written sentences were classified as High Confidence LLM-written, which was important to me considering how disastrous being accused of using LLMs in writing can be. But the False Negative amount was unsatisfactory.

**Works Cited**

[1] Z. Durumeric and H. W. A. Hanley. “Machine-Made Media: Monitoring the Mobilization of Machine-Generated Articles on Misinformation and Mainstream News Websites”. Available: <https://arxiv.org/abs/2305.09820>. [Accessed Feb. 24, 2025].

[2] J. Wu, S. Yang, R. Zhan, and Y. Yuan. “A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions”. Available: <https://direct.mit.edu/coli/article/doi/10.1162/coli_a_00549/127462>. [Accessed Feb. 24, 2025].

[3] J. Rudolph, S. Tan, and S. Tan. “War of the chatbots: Bard, Bing Chat, ChatGPT, Ernie and beyond. The new AI gold rush and its impact on higher education”. Available: https://journals.sfu.ca/jalt/index.php/jalt/article/view/771/577. [Accessed Feb. 24, 2025].

[4] J. F. O’Brien. “Accusatory AI: How a Widespread Misuse of AI Technology Is Harming Students”. Available: https://medium.com/towards-data-science/accusatory-ai-how-misuse-of-technology-is-harming-students-56ec50105fe5. [Accessed Feb. 24, 2025].

[5] H. Q. Nguyen-Son, M. S. Dao, and K. Zettsu. “SimLLM: Detecting Sentences Generated

by Large Language Models Using Similarity

between the Generation and its Re-generation”. Available: https://aclanthology.org/2024.emnlp-main.1246.pdf. [Accessed Feb. 24, 2025].

[6] X. Yang, W. Cheng, Y. Wu, L. Petzold, W. Y. Wang, and H. Chen. “DNA-GPT: Divergent N-Gram Analysis for Training-Free Detection of GPT-Generated Text”. Available: <https://arxiv.org/abs/2305.17359>. [Accessed Feb. 24, 2025].

[7] C. Mao, C. Vondrick, H. Wang, and J. Yang. “Raidar: geneRative AI Detection viA Rewriting”. Available: <https://arxiv.org/abs/2401.12970>.[Accessed Feb. 24, 2025].

[8] J. Wu, R. Zhan, D. F. Wong, S. Yang, X. Liu, L. S. Chao, and M. Zhang. ”Who Wrote This? The Key to Zero-Shot LLM-Generated Text Detection Is GECScore”. Available: <https://arxiv.org/abs/2405.04286>**.** [Accessed Feb. 24, 2025].