Detect AI Generated Text with Large Language Model

Steven Wen
McGill University
260887963
yifan.wen@mail.mcgill.ca

Haikun Zhao
McGill University
260846475
haikun.zhao@mail.mcgill.ca

Abstract

In the era of advancing artificial intelligence, Large Language Models (LLMs) like GPT-4 have revolutionized natural language processing, generating text that is difficult to distinguish from human-written text, posing significant challenges in distinguishing AI-generated content from genuine human creations. In this project we create a detection algorithm to accurately detect whether an essay was written by a student or an LLM. Our methodology built up on SeqXGPT to enhance detection accuracy beyond current state-of-the-art solutions. The importance of this research is to maintain the value of human creativity, reduce chances that humans will be replaced or alter work. Enhance cybersecurity by identifying potential vulnerabilities in networks and systems and detecting malicious activities. Preliminary findings from our research show promising results in effectively distinguishing between human and AI-authored texts.

1 Introduction

In recent years, the progress of Large Language Models (LLMs) such as GPT, Llama, and Claude has significantly advanced the field of Natural Language Processing (NLP). These developments have enhanced capabilities in Natural Language Understanding (NLU) and Natural Language Generation (NLG), leading to improvements across various NLP tasks. These tasks include text summarization, text classification, grammar correction, automated answering, and text completion, among others. By engineering prompts those LLMs have a significant improvement than the earlier models like BERT, RoBERTa, and DistilBERT on both accuracy and distinguishability from human text.

However, this rapid technological progress introduces significant challenges, particularly in detecting AI-generated text. It is critical for maintaining the authenticity and regulations. This capability opens the door for people to use it and be misused for unethical purposes.According to (Heikkilä, 2022), these applications range from plagiarism and the generation of fake news to spamming, crafting deceitful reviews, and manipulating web content for nefarious ends.(Christian, 2023) It also affects cybersecurity. The ability of LLMs to produce convincing and seemingly authentic text can be exploited to carry out phishing attacks and undermine digital security on a broad scale.(Krishnan, 2023) Therefore, promoting the responsible use of generative AI tools and identifying a generated text will be a challenging task.

Thus for this research we plan to explore different methods to identify distinguishable features and patterns of generated text data. While existing solutions have been made to identify machine-generated text, they often rely on patterns that can be mimicked or obscured by newer, more advanced models. Traditional methods focus on detecting repetitive phrases or unusual syntax. In light of these challenges, our research aims to develop methods that can overcome the limitations of current models.

This paper explores the capabilities of current Large Language Models (LLMs) in the context of detecting AI-generated text within the field of natural language processing. The choise of choosing this problem is driven by the growing need to maintain the authenticity and integrity of digital information in an era increasingly dominated by AI. Despite the widespread availability of tools and techniques for text analysis, we want to provide a more robust solution to the problem. We will further analyze the impact of our approach in subsequent sections, delving into the broader implications and the practical applicability of our detection methods in high levels of accuracy and reliability.

2 Related Work

SeqXGPT: Sentence-Level AI-Generated Text Detection

The SeqXGPT study(Wang et al., 2023) represents a groundbreaking approach in targeting sentence-level detection challenges. Unlike conventional document-level analyses, this methodology harnesses log probability lists derived from white-box Large Language Models as unique features for discrimination. SeqXGPT employs convolution and self-attention networks to analyze these features, which enables the model to discern subtle distinctions between human-written and AI-generated sentences, showcasing a significant leap in detection capabilities within mixed-origin documents. The innovation lies not just in the accuracy of detection but also in the sophisticated use of LLMs' internal outputs as a data source for identifying generative patterns specific to AI-created content.

Building upon the foundation laid by SeqXGPT, our proposed approach aims to extend these capabilities through the integration of dynamic feature extraction techniques and the inclusion of broader contextual analyses. We plan to develop an adaptive model that not only leverages log probability features but also incorporates additional linguistic and semantic cues extracted from the text. This would involve employing advanced machine learning algorithms capable of identifying complex patterns and dependencies that signify AI generation, such as GNNs.

ConDA: Contrastive Domain Adaptation for AI-generated Text Detection

The ConDA study(et al., 2023a) introduces a technique focusing on the issue of scarce labeled data. This scarcity is a significant bottleneck in training supervised detectors capable of identifying AI-generated content accurately. ConDA addresses this challenge by framing AI text detection as an unsupervised domain adaptation task. Given the rapid development of new Large Language Models, the subsequent diversity in AI-generated texts may not always have corresponding labeled datasets for training. ConDA combines standard domain adaptation techniques with the robust representation capabilities of contrastive learning to learn domain-invariant features that can accurately classify text as AI-generated or human-written, regardless of the specific LLM it originated from.

Inspired by ConDA's approach, our proposed methodology aims to further explore and expand the boundaries of domain adaptation and contrastive learning in AI text detection. We plan to integrate a wider array of LLMs and diversify the unsupervised adaptation strategies to encompass newer, more complex models of text generation. Additionally, we will investigate enhancing the contrastive learning aspect by incorporating multi-modal data sources, potentially improving the model's generalization capabilities across various AI-generated content forms.

DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature

The DetectGPT(et al., 2023c) study proposes an innovative method for identifying AI-generated text. This approach leverages the probability curvature of text generated by language models to distinguish it from human-written content. By analyzing the regions of negative curvature in a model's log probability function, DetectGPT can identify machine-generated texts without the need for additional training or specific datasets. This zero-shot detection capability offers a practical tool for distinguishing between human and AI-generated content efficiently.

Building on the curvature-based approach of DetectGPT, our proposed method aims to incorporate linguistic features such as syntactic complexity, coherence measures, and semantic anomaly detection. By analyzing sentence structures, coherence between consecutive sentences, and the semantic appropriateness of content within the context, we aim to enhance the model's sensitivity to subtle cues of AI generation, and thus to provide a multidimensional zero-shot framework for AI text detection.

Stacking the Odds: Transformer-Based Ensemble for AI-Generated Text Detection

This study(et al., 2023b) exemplifies an advanced approach to identifying AI-generated text using ensemble methods, specifically leveraging Transformer models. The study underscores a strategic ensemble of accessible and lightweight Transformer models, aimed at enhancing detection performance without the high computational costs typically associated with these architectures. This ensemble approach leverages the unique strengths of each constituent model to improve overall accuracy. The benefit of this approach lies in its ability to harness the diversity of predictions from multiple models, thereby reducing the likelihood of

misclassification and significantly boosting detection rates.
To make use of this ensemble methodology, our proposed approach seeks to advance the concept further by integrating broader machine learning models into the ensemble, beyond Transformers. We aim to utilize different architectural strengths, including those of non-Transformer models like CNNs and RNNs, to create a more robust ensemble system capable of handling the variability of AI-generated texts. Based on the context and characteristics of the text being analyzed, we incorporate a dynamic weighting mechanism that adaptively adjusts the influence of each model in the ensemble to enhance the system's adaptability across a wider range of AI-generated content scenarios.

3 Modeling

3.1 Baseline

3.1.1 SeqXGPT

For our baseline model we have used the Sequence X(Check) GPT (SeqXGPT)(Wang et al., 2023), a sequence-to-sequence transformer model. The model is using log probability lists from white-box LLMs as distinctive features, treated analogously to wave patterns in speech processing and are undetectable by LLMs themselves. SeqXGPT combines convolution and self-attention networks to analyze these features.
The structure of SeqXGPT is shown in Fig1 which consists of the following three parts:

1. Perplexity Extraction and Alignment: This layer is feature extraction which involves computing the perplexity scores for sentences or text fragments using a white-box LLM. Perplexity measures how well a probability model predicts a sample. For each word or token in the sequence, a log probability is obtained, which representing the foundational features.

2. Feature Encoder: This layer consists of convolutional neural networks (CNNs) and transformers. The CNNs are capable of identifying local patterns within the perplexity scores and can extract higher-level feature representations from the data. The transformer helps in capturing the contextual relationships within the input sequence through self-attention mechanisms.

3. Linear Classification Layer: The last layer of the model uses linear classifiers that operate on the high-level features provided by the CNNs. Each classifier outputs a prediction of whether the text is human written or generated by LLM.

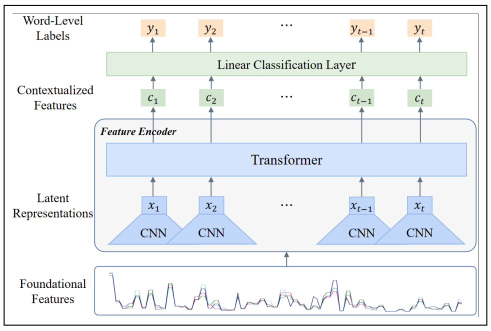

Figure 1: SeqXGPT Architecture.(Wang et al., 2023)

We have chosen this as the baseline since this model has shown improvement on DetectGPT and Sent-RoBERTa. Thus we want to see by changing some structure of the SeqXGPT would it make it out perform the current models.

3.1.2 SeqXGPT with RoBERTa

Since CNNs are generally used for capturing local and positional patterns we want to see if using a model that is more suitable for natural language processing (NLP) tasks would increase the performance of the model. Thus we have used RoBERTa(Liu et al., 2019) as a replacement for CNNs. Also since RoBERTa has pretrained on a large corpus of text it would need less task specific tuning that CNNs require.

3.2 Our Proposed Model

The proposed model for detecting AI-generated text employs a sophisticated multi-layered architecture that leverages advanced techniques in natural language processing. At the core of this model is the RoBERTa base model. RoBERTa's ability to understand deep contextual differences makes it an excellent choice for the foundational layer, where it processes the input text to generate a base representation for further analysis.
To enhance the stability and efficiency of training, layer normalization is applied immediately after the RoBERTa output. This normalization method helps stabilize the learning process and is critical for deep network architectures, as it ensures faster convergence and reduces sensitivity to network initialization. Following this, a multihead attention

mechanism is utilized to allow the model to focus on relevant parts of the text. This layer is crucial as it processes the information through multiple attention heads, capturing various semantic and contextual relationships that are indicative of AI-generated text.

The model also incorporates residual connections following the attention and convolutional layers. These connections help preserve information throughout the network and mitigate the vanishing gradient problem, which is vital for effectively training deeper neural network structures. The convolutional layer further processes the text to capture local feature dependencies, providing the ability to detect syntactic patterns specific to AI-generated or human-generated text.

Another sophisticated element in the model is the use of Gated Linear Units (GLU). This layer employs a gating mechanism to control the flow of information, allowing the model to capture more complex data patterns. GLUs are particularly effective in modeling non-linear relationships, which are essential for identifying subtle differences in the text generated by AI.

The architecture also features a transformer encoder and decoder. The encoder learns deeper contextual relationships within the text, while the decoder focuses on synthesizing relevant information for classification. This dual mechanism enhances the model's capability to thoroughly analyze and interpret the text, ensuring high accuracy in distinguishing between human and AI-generated content.

Finally, a position-wise feed-forward network and a classifier are used to transform and finalize the model's output. The feed-forward network applies additional transformations to capture non-linear relationships at each position independently, enriching the model's understanding of the text. The classifier then maps these features to the output classes, providing the necessary decision-making capability to classify the text as either real or AI-generated.

Overall, the integration of these advanced techniques and layers forms a robust model capable of effectively detecting AI-generated text. Each component is chosen not only for its individual merits but also for how it complements the other parts of the architecture, resulting in a comprehensive approach that addresses the complexities and subtleties of natural language processing and AI-generated text detection.

4 Dataset and evaluation

4.1 Dataset

4.1.1 section{test|train}-essays.csv

The first dataset (Kaggle, 2023) we will be using is comprises about 10,000 essays, some written by students and some generated by a variety of large language models (LLMs). The goal of the competition is to determine whether or not essay was generated by an LLM. All of the essays were written in response to one of seven essay prompts. In each prompt, the students were instructed to read one or more source texts and then write a response. This same information may or may not have been provided as input to an LLM when generating an essay.

- `id` - A unique identifier for each essay.
- `prompt_id` - Identifies the prompt the essay was written in response to.
- `text` - The essay text itself.
- `generated` - Whether the essay was written by a student (`0`) or generated by an LLM (`1`). This field is the target and is not present in `test_essays.csv`.

4.1.2 Mistral7B_CME_v6.csv

The second dataset(Ellis, 2023) we will be using contains 4,900 essays that were produced by LLM.

- `prompt_id` - Identifies the prompt the essay was written in response to.
- `prompt_name` - The name of the prompt given to the LLM.
- `text` - The essay text itself.
- `generated` - Since this dataset was all LLM generated thus this is all 1.

From the first dataset we find that the train_essays_df DataFrame reveals a highly imbalanced distribution with 1,375 student-written essays (label '0') and only 3 LLM-generated essays (label '1'). That is why we decided to merge the two dataset together to have a more balanced dataset. And since we are focusing on detecting if the text is human written or generated by LLM, we decided to only keep the text and generated columns for training.

4.2 Evaluation Metric

During the training process, a cross-entropy loss function measures how well the model's predicted probabilities agree with the actual labels. This loss provides direct feedback to the optimizer for adjusting the model weights to improve performance. The primary metric used is accuracy, which calculates the proportion of correct predictions made by the model over the total number of predictions. This metric is straightforward and gives a quick overview of how well the model is performing overall. However, accuracy alone might not always provide a complete picture, especially in cases where the data distribution is imbalanced between classes. To address potential imbalances and to gain deeper insights into the model's performance, precision and recall are also calculated. Precision measures the accuracy of positive predictions made by the model, quantifying its ability to not label as positive a sample that is negative. On the other hand, recall assesses the model's ability to find all the relevant cases within a dataset. These metrics are crucial for understanding the trade-offs between catching as many positives as possible (high recall) and ensuring that the positives caught are indeed correct (high precision).

5 Experimental details

5.1 Training

The model underwent a structured training regimen over 10 epochs on GPU. The data was processed in mini-batches, with each batch being fed through the model to compute outputs and the associated loss using a predefined loss function.

Gradient Accumulation:

To manage the computational load and optimize the memory usage, we implemented gradient accumulation over 4 steps. This technique allows us to effectively handle larger batches of data by accumulating gradients over multiple mini-batches, thus simulating the effect of training with larger batches.

Loss Normalization and Backpropagation:

Each loss computed was normalized by the accumulation steps to ensure consistent updates across different batch sizes. Following the loss computation, backpropagation was performed to compute the gradients. To mitigate the risk of exploding gradients, we applied gradient clipping with a maximum norm of 1.0.

Parameter Update:

The optimizer updated the model parameters only after every fourth batch, post-accumulation, to reflect the accumulated gradients, followed by resetting the optimizer's state.

Learning Rate Scheduler:

An exponential learning rate scheduler (ExponentialLR) was used, with a decay factor (gamma) of 0.8, to adjust the learning rate after each epoch. This adjustment helps in fine-tuning the learning process by reducing the learning rate as the training progresses, aiding in the convergence towards the global minimum of the loss function.

Performance Monitoring:

Throughout the training phase, the progress and performance were monitored using a progress bar (tqdm). After each epoch, the average loss and accuracy were computed and displayed, providing real-time feedback on the training effectiveness.

5.2 Validation

After each training epoch, the model was evaluated in a separate validation phase to assess its performance on unseen data, which is crucial for detecting potential overfitting and tuning the model's hyperparameters.

The model was switched to evaluation mode to disable certain operations like dropout and batch normalization specific to the training phase. The validation data was processed in batches, similar to the training phase, but without performing any backpropagation or parameter updates. Loss and accuracy for each batch were recorded to calculate the overall validation loss and accuracy, providing insights into how well the model generalized to new data.

5.3 Testing

After the model has been trained, a final test phase was conducted to evaluate the model's performance on an independent test dataset. Similar to the validation phase, the model was set to evaluation mode to ensure consistency in the way the model processes the data. The test data was processed in batches, and accuracy was calculated for each batch. These accuracy measures were then aggregated to determine the final accuracy score, which serves as the definitive assessment of the model's capability to detect AI-generated text accurately.

6 Results and Discussion

6.1 Detection Accuracy

We evaluated the performance of our model compared to the baseline SeqXGPT model on a balanced dataset comprising both human-written and AI-generated texts. The results are as shown in Table 1.

Model	Accuracy (%)
Our Model	99.1
SeqXGPT	98.4

Table 1: Accuracy of each model on the test dataset.

Our model achieved an accuracy of 99.1%, slightly outperforming the SeqXGPT model, which recorded an accuracy of 98.4%. These results are indicative of the effectiveness of the enhancements made in our approach, specifically the integration of sophisticated feature extraction mechanisms and advanced machine learning techniques.

6.2 Time and Complexity

In addition to achieving higher accuracy, our model demonstrated significant improvements in terms of computational efficiency. The training and inference times were considerably lower than those of the SeqXGPT model. This reduction can be attributed to the optimized architecture of our model, which integrates RoBERTa with advanced neural network techniques, enhancing the processing speed without compromising the accuracy of text classification.

6.3 Discussion

The slight increase in accuracy with our model suggests that the integration of RoBERTa and additional feature analysis layers effectively captures the subtle variations that differentiate human-written texts from AI-generated texts. The ability of RoBERTa to understand deeper contextual differences and the use of a multi-head attention mechanism likely contributed to these improvements.

Furthermore, the efficiency gains highlight the potential for deploying this model in real-time applications where processing speed is critical, such as live monitoring of content on social media platforms.

However, there are several limitations to consider. The model's performance, while better in our testing environment, may vary when exposed to texts generated by newer or different types of AI models not represented in the training data. Future work should focus on continually updating the model with data from newer AI text generators to maintain its effectiveness.

Additionally, while the model performs well in a controlled testing environment, its real-world effectiveness could be influenced by factors such as changes in language usage over time or the introduction of novel slang and expressions, which might not be immediately reflected in the training dataset.

Overall, the results are promising, and the improvements in detection accuracy and processing efficiency suggest that our model could be a valuable tool for identifying AI-generated text in various applications, from academic integrity to cybersecurity.

7 Conclusion

This research explores AI-generated text detection through a comprehensive examination of several studies, each employing distinct methodologies to address the challenge of differentiating between human and machine-generated content. SeqXGPT focuses on sentence-level detection using log probability features derived from large language models.

Our methodology is influenced by these innovative strategies, seeking to amalgamate their strengths while introducing significant refinements. It involves the integration of dynamic feature extraction, expansion of domain adaptation strategies, enhancement of linguistic features, and the inclusion of a varied set of machine learning models within an ensemble architecture. The objective is to devise a detection system that is more resilient, flexible, and efficient. This approach not only builds upon the successes of existing methods but also endeavors to extend the current capabilities in AI-generated text discrimination across different contexts and generators. Ultimately, this work aims to contribute to the field's progression by developing a more resilient, adaptable, and efficient system for detecting AI-generated content.

8 Code

The code can be find on GitHub. The instruction in under README.

References

Jon Christian. 2023. Cnet secretly used ai on articles that didn't disclose that fact, staff say. futurism. *Futurism*.

Carl McBride Ellis. 2023. Llm mistral 7b instruct texts dataset. `https://www.kaggle.com/datasets/carlmcbrideellis/llm-mistral-7b-instruct-texts?select=Mistral7B_CME_v7.csv`. Accessed: 2024-04-12.

Amrita Bhattacharjee et al. 2023a. Conda: Contrastive domain adaptation for ai-generated text detection. *Papers With Code*.

Duke Nguyen et al. 2023b. Stacking the odds: Transformer-based ensemble for ai-generated text detection. *Papers With Code*.

Eric Mitchell et al. 2023c. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *Papers With Code*.

Melissa Heikkilä. 2022. How ai-generated text is poisoning the internet. *MIT Technology Review*.

Kaggle. 2023. Dataset for llm-detect ai-generated text competition. `https://www.kaggle.com/competitions/llm-detect-ai-generated-text/data?select=train_essays.csv`. Accessed: 2024-04-12.

Ashwin Krishnan. 2023. Generative ai is making phishing attacks more dangerous: Techtarget. *TechTarget*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023. Seqxgpt: Sentence-level ai-generated text detection.

A Appendix

A.1 Evaluation

A.1.1 Accuracy

We will valuated on area under the ROC curve between the predicted probability and the observed target.

A.2 Contributions

Steven Wen: data collection and pre-processing. Constructing two baseline models. Haikun Zhao: Model researching. Construct our proposed model. The report was divided evenly among the team.