PRDetect: Perturbation-Robust LLM-generated Text Detection Based on Syntax Tree

Anonymous ACL submission

Abstract

As LLM-generated text becomes increasingly prevalent on the internet, which may contain hallucinations or biases, detecting such content has emerged as a critical area of research. Recent methods have demonstrated impressive performance in detecting text generated entirely by LLMs. However, in real-world scenarios, users often make perturbations on the LLM-generated text, and the robustness of existing detection methods to these perturbations has not been sufficiently explored. This paper empirically investigates this question and finds that even minor perturbation can severely degrade the performance of current detection methods. To address this issue, we find that the syntactic tree is minimally affected by disturbances and exhibits differences between human-written text and LLM-generated text. Therefore, we propose a detection method based on syntactic trees, which can capture features invariant under perturbations. It demonstrates significantly improved robustness against perturbation on the HC3 and GPT-3.5-mixed datasets.

1 Introduction

The proliferation of LLM-generated texts on the internet has raised numerous issues, such as fake news (Zellers et al., 2020) and papers. which is difficult to identify (Gehrmann et al., 2019). In recent years, the task of LLM-generated text detection has also shown good results on LLM-generated texts (Mitchell et al., 2023; Liu et al., 2023; Bao et al., 2024; McGovern et al., 2024).

However, we argue that the previous task settings were overly simplistic, making it hard to reflect real-world scenarios where LLM-generated text is frequently modified and adjusted. This paper finds that if the text is subjected to certain perturbations, the effectiveness of many detection tools will drop significantly, as depicted in Figure 1.

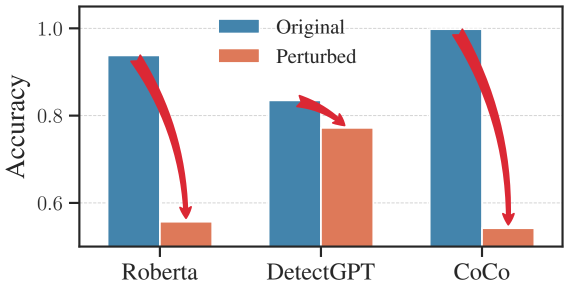

Figure 1: The accuracy of several detection methods drop significantly after perturbing just 10% words of each LLM-generated sentence in the HC3 datasets.

To solve this problem, we find differences in the syntax trees between human-written texts and LLM-generated texts shown in Section 4.2, which exhibit minimal susceptibility to perturbations. Based on this finding, this paper presents a perturbation-robust text detection method (PRDetect) and proposes a perturbation method that mimics human editing. Under perturbation, the paper compares the PRDetect with several well-performing baselines. PRDetect demonstrates both high accuracy and perturbation-robustness.

In summary, our contributions are as follows:

- PRDetect leverages the differences in syntax trees and demonstrates outstanding performance on two datasets of different lengths.
- We propose a novel perturbation method to emulate the processes of real-world text polishing.
- PRDetect possesses state-of-the-art perturbation robustness.

2 Related Work

2.1 LLM-Generated Text Detection

The task of detecting LLM-generated texts is distinguishing whether a piece of text is written by humans or generated by LLMs. Existing methods can be broadly categorized into four groups.

Featured-based text detection. The various features within a text can be employed to train a model for classification. GLTR (Gehrmann et al., 2019) calculates three features for detection: the probability of the next word, the absolute rank of the next word, and the entropy of the predicted distribution. LLMDet (Wu et al., 2023) saves a local probability dictionary to calculate perplexity for classification, which can save storage space. CoCo (Liu et al., 2023) uses entity graphs for detection, which performs well in detecting long texts.

Fine-tuning large pre-trained model. Pre-trained language models offers significant advantages in NLP tasks, which eliminates the need for manually specified features. Transformer-based models can be used to distinguish whether a piece of text was generated by ChatGPT or manually (Mitrović et al., 2023). OpenAI fine-tuned a RoBERTa model[1] to detect text generated by GPT-2. These methods can be further refined and enhanced by fine-tuning with local data.

Zero-shot method. It relies on certain statistical regularities, saving time in training the model. DetectGPT (Mitchell et al., 2023) found machine-generated text tends to occupy regions of negative curvature in the model's log-probability function. Perturb the text and calculate the changes in log probability. Those with smaller average changes are more likely to be human-written texts. Fast-DetectGPT (Bao et al., 2024), DetectGPT-SC (Wang et al., 2023) and DetectGPT4Code (Yang et al., 2023) also achieved zero-shot classification.

Text watermarking method. Adding a watermark involves embedding a hidden representation into the text, which is difficult for humans to detect or eliminate. Such as selecting words from the green list (Kirchenbauer et al., 2023), generating a private key to create a watermark (Kirchenbauer et al., 2023), using neural networks for watermark generation and detection (Liu et al., 2024a).

2.2 Text Perturbation Analysis

Existing experiments have shown that simple perturbations can significantly interfere with detectors, such as replacing characters with visually similar letters from different languages (Wolff and Wolff, 2022), swapping letters within words (Huang et al., 2024), back-translation or rewriting (Macko et al., 2024). Some papers have conducted perturbation experiments at the token-level on the text (Liu et al., 2023).

3 Methods

The primary framework of PRDetect comprises the construction of syntax trees, node encoding, supervised training of a graph convolutional network, and text perturbation for testing purposes. The main process is depicted as shown in Figure 2.

3.1 Syntax tree construction and node encoding

In this paper, we utilize spaCy[2] and Roberta to accomplish this process.

For a given long input text, we utilize spaCy for tokenization after segmenting the text into chunks. SpaCy performs part-of-speech tagging on each token and determines the dependency relationships using a set of rules, such as identifying the subject-verb relationship or the modifying relationships. Using this method, a dependency tree is constructed, where each node represents a token. Subsequently, we construct an adjacency matrix A based on the dependency tree, 1 is used to represent dependency relationship between two tokens, and 0 otherwise.

For the token nodes of the dependency tree, we use Roberta to get their embedding, which are used to initialize the nodes in the graph network. Compared to random initialization, this approach can lead to faster convergence and improved performance.

At this point, we have obtained the adjacency matrix, the embeddings for the nodes and the text labels for the training network.

3.2 Graph Convolutional Network

In this paper, we utilize two layers of Graph Convolutional Network(GCN) (Kipf and Welling, 2017) to perform graph convolution operations. Each layer of the convolution can be expressed as:

$$H^{(l)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l-1)}) \quad (1)$$

where $H^{(l)}$ is the node embedding matrix at layer l, $\hat{A}$ is the adjacency matrix of the graph that incorporates self-loops, $\hat{D}$ is the diagonal degree matrix, W^l is the weight matrix for layer l, and σ is a non-linear activation function, typically ReLU.

[1] https://github.com/openai/gpt-2-output-dataset/tree/master/detector

[2] https://github.com/explosion/spaCy

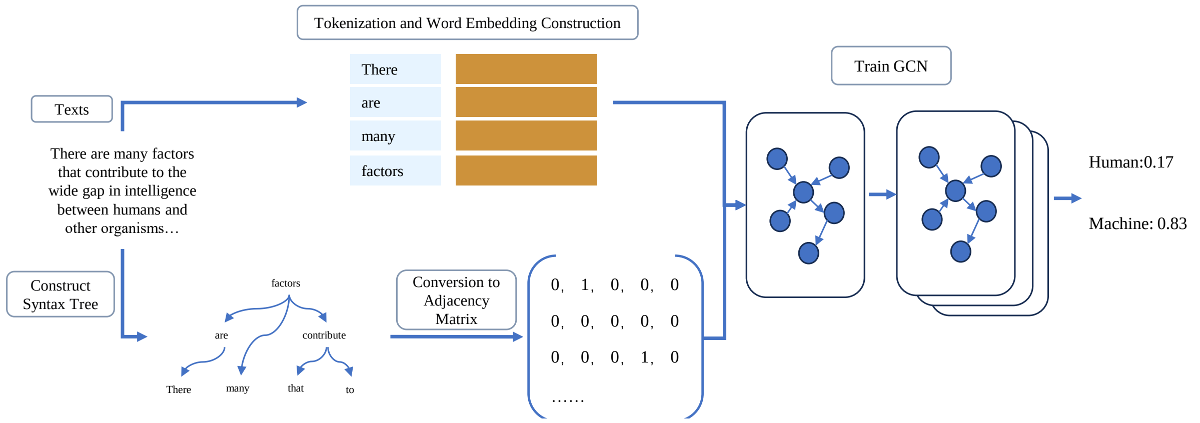

Figure 2: The primary procedure of PRDetect. It constructs and encodes syntax trees and nodes to train a GCN for text detection.

Dataset	HC3		GPT3.5-Mixed	
	Human	Machine	Human	Machine
Depth of Nodes	2.80	3.26	3.13	3.15
Number of Nodes	20.23	25.34	25.08	25.09
Height of Root	4.79	6.38	5.61	6.18
Length of Text	147.93	178.65	756.55	501.13

Table 1: Statistical analysis of dataset. The values in the table are all averages.

Self-loops for nodes can reinforce the inherent features of the nodes during the convolution process, represented as:

$$\hat{A} = A + I \tag{2}$$

where I is the identity matrix of the same dimension as A. Our model employs the Binary Cross-Entropy Loss as the loss function L, which is suitable for the binary classification task.

4 Dataset and Syntactic Tree Difference Analysis

4.1 Datasets and Metrics

The text generation capabilities of LLMs can affect the difficulty of text detection tasks. We choose texts generated by more recent LLMs, which are typically more fluent and difficult for humans to directly distinguish from human-written text.

Human ChatGPT Comparison Corpus (HC3)[3] (Guo et al., 2023). In this dataset, there are questions and answers from ChatGPT and human experts, spanning various domains such as computer science, finance, medicine, law, and psychology.

GPT3.5-Mixed[4] (Liu et al., 2023). This dataset is generated by text-davinci-003, focusing on the news domain. The texts included are longer compared to those in the HC3 dataset. The Mixed dataset includes 17 different sources, such as news websites like CNN, BBC, and Yahoo.

Following several related works (Wu et al., 2023; Liu et al., 2023), we use **accuracy** and **F1 score** as metrics.

4.2 Syntactic Tree Difference Analysis

We conduct a statistical analysis of human-written texts and LLM-generated texts in the HC3 and GPT3.5-Mixed.

Table 1 presents the average number of nodes in the syntax tree, the average height of the root node, the average depth of nodes per tree, and the average length of the texts in the dataset. The first three are some basic characteristics of the graph structure. It can be observed that, aside from the average number of nodes in the GPT3.5-Mixed dataset, other features show noticeable differences between human-written and LLM-generated texts. This allows the GCN to learn these differences and classify correctly. Furthermore, the difference in length between the two datasets also has a certain impact in the experiments, shown in the Appendix B. Detailed analysis and distribution graphs can be found in the Appendix C.

[3]https://github.com/Hello-SimpleAI/chatgpt-comparison-detection

[4]https://huggingface.co/datasets/ZachW/MGTDetect_CoCo

Dataset	HC3					GPT3.5-Mixed				
Ratio	0%	5%	10%	20%	30%	0%	5%	10%	20%	30%
RoBERTa	0.9380	0.5800	0.5570	0.5270	0.5080	0.8927	0.5055	0.4995	0.4945	0.4945
DetectGPT	0.8350	0.8010	0.7720	0.7030	0.6580	0.6060	0.5860	0.5820	0.5680	0.5500
CoCo	**0.9981**	0.5432	0.5421	0.5356	0.5333	**1.0000**	0.6995	0.6893	0.6829	0.6805
PRDetect	0.9850	**0.9870**	**0.9880**	**0.9890**	**0.9880**	0.9610	**0.9570**	**0.9570**	**0.9590**	**0.9610**

Table 2: Accuracy of different models on LLM-generated texts and perturbed texts. CoCo demonstrated the best performance on original texts. PRDetect showed the highest overall effectiveness, exhibiting state-of-the-art performance on perturbed texts.

5 Experiments

5.1 Baselines

In our study, we compared PRDetect with several state-of-the-art detectors designed for LLM-generated text identification.

RoBERTa (Liu et al., 2019) is an advanced NLP model that improves upon BERT (Devlin et al., 2018). In this paper, we employ a version of RoBERTa that has been fine-tuned by OpenAI[5].

DetectGPT (Mitchell et al., 2023) is a zero-shot LLM-text detection method.

CoCo (Liu et al., 2023) leverages entity graph for training a text detection model.

5.2 Text Perturbation

While constructing the syntax tree in subsection 3.1, we obtain the part of speech for each word. Selecting a category of words for marking. Then, we employ WordNet, which is part of the NLTK[6], to obtain a list of synonyms for a word. From this list, we opt to replace the original word with the first synonym listed.

When we replace the synonyms for adjectives, we select proportions of 5%, 10%, 20%, and 30%. The paper also compares other word-level perturbations in Appendix A.

5.3 Main Experiments

We primarily compared the detection accuracy of PRDetect with other baselines on both LLM-generated texts and perturbed texts.

Detecting LLM-generated texts. The results of our experiments are detailed in Table 2. PRDetect achieved an accuracy rate of 98.5% on the HC3 dataset and 96.1% on the GPT3.5-Mixed dataset, demonstrating its effectiveness in detecting LLM-generated text. Both PRDetect and CoCo, which utilized GCN to learn graph features, outperformed the other two methods based on semantic features, which proves the effectiveness of graph information in detecting text. DetectGPT faces challenges in detecting long texts, which is an issue noted on its official Github[7].

Detecting perturbed texts. We perturbed the test set texts according to the method described in Section 5.2. Table 2 demonstrates that PRDetect achieves the highest detection accuracy for perturbed texts. Moreover, as the degree of perturbation varies, the accuracy of PRDetect declines by no more than 0.05%. In contrast, the other baselines experience a decrease in accuracy as the perturbation intensity increases.

In summary, PRDetect demonstrates a strong capacity to resist text perturbation while maintaining a high detection accuracy rate. In Section A, we will further compare them with other perturbation methods, showcasing PRDetect's perturbation robustness.

6 Conclusion

In this paper, we propose PRDetect, a perturbation-robust detection method for LLM-generated text, which leverages differences in syntax trees to train a GCN. Not only can it effectively identify generated text, but also possesses strong perturbation robustness. To mimic the polishing of generated text before its actual use, we propose a perturbation method based on synonym replacement. PRDetect is minimally affected by text perturbation on the HC3 and GPT3.5 datasets and its accuracy is significantly higher than that of other baselines.

[5] https://github.com/openai/gpt-2-output-dataset/tree/master/detector

[6] https://github.com/nltk/nltk

[7] https://github.com/eric-mitchell/detect-gpt/issues/4

Limitations

Although PRDetect demonstrates robustness against perturbations, there are still some imperfections that need to be addressed.

The types of perturbations. The text perturbations discussed in this paper are all the token-level. We have not tested methods such as backtranslation and rewriting at the sentence-level. There are two reasons: First, sentence-level perturbations have a significant impact on the graph structure, making it difficult to detect using the approach of this paper. Second, it is challenging to specify the proportion of perturbation at the sentence-level, and texts with perturbations exceeding 50% are difficult to label. The issue of sentence-level perturbations requires further definition and analysis.

Different Length and Cross-Dataset Detection. Short text detection remains a challenge for most classifiers. As shown in the Appendix B, the performance of PRDetect, when trained on long texts, significantly declines when the text length falls below 300 characters, with accuracy levels between 0.6 and 0.75. However, when trained on the short text dataset HC3, the performance drop is not as pronounced. Furthermore, we have observed that model trained with short texts achieves an accuracy of 0.87 when detecting long texts. Conversely, when model trained on long texts is used to detect short texts, the accuracy is only 0.73. The specific reasons behind this discrepancy are yet to be discovered.

References

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. *Preprint*, arXiv:2310.05130.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. Gltr: Statistical detection and visualization of generated text. *Preprint*, arXiv:1906.04043.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *Preprint*, arXiv:2301.07597.

Guanhua Huang, Yuchen Zhang, Zhe Li, Yongjian You, Mingze Wang, and Zhouwang Yang. 2024. Are ai-generated text detectors robust to adversarial perturbations? *Preprint*, arXiv:2406.01179.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Preprint*, arXiv:1609.02907.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *Preprint*, arXiv:2301.10226.

Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2024a. An unforgeable publicly verifiable watermark for large language models. *Preprint*, arXiv:2307.16230.

Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. 2024b. Does detectgpt fully utilize perturbation? bridge selective perturbation to fine-tuned contrastive learning detector would be better. *Preprint*, arXiv:2402.00263.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2023. Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. *Preprint*, arXiv:2212.10341.

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. *Preprint*, arXiv:1907.11692.

Dominik Macko, Robert Moro, Adaku Uchendu, Ivan Srba, Jason Samuel Lucas, Michiharu Yamashita, Nafis Irtiza Tripto, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2024. Authorship obfuscation in multilingual machine-generated text detection. *Preprint*, arXiv:2401.07867.

Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. 2024. Your large language models are leaving fingerprints. *Preprint*, arXiv:2405.14057.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *Preprint*, arXiv:2301.11305.

Sandra Mitrović, Davide Andreoletti, and Omran Ayoub. 2023. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *Preprint*, arXiv:2301.13852.

Rongsheng Wang, Qi Li, and Sihong Xie. 2023. Detectgpt-sc: Improving detection of text generated by large language models through self-consistency with masked predictions. *Preprint*, arXiv:2310.14479.

Max Wolff and Stuart Wolff. 2022. Attacking neural text detectors. *Preprint*, arXiv:2002.11768.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. Llmdet: A third party large language models generated text detection tool. *Preprint*, arXiv:2305.15004.

Xianjun Yang, Kexun Zhang, Haifeng Chen, Linda Petzold, William Yang Wang, and Wei Cheng. 2023. Zero-shot detection of machine-generated codes. *Preprint*, arXiv:2310.05103.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2020. Defending against neural fake news. *Preprint*, arXiv:1905.12616.

A Other Perturbation Types

Model	Original	Insert	Repeat	Replace	Detele
CoCo	0.9981	0.4733	0.5380	0.4713	0.5212
PRDetect	0.9850	0.9830	0.9820	0.7980	0.7470

Table 3: Accuracy on four common types of perturbation.

In some papers (Liu et al., 2023, 2024b), they randomly **insert**, **delete**, **repeat** or **replace** words to perturb the text. We applied these four types of perturbations at 25% ratio on the HC3 test dataset. As shown in Table 3, PRDetect is hardly affected by Insert and Repeat perturbations, as these modifications have minimal impact on the original syntax tree. For the other two methods that alter the syntax tree, the detection accuracy of PRDetect declines but still maintains good performance.

B Short Text Detection

Length	Original	[300, 400)	[200, 300)	[150, 200)	[100, 150)
Acc	0.9610	0.7450	0.7575	0.6900	0.6200
F1	0.9617	0.7571	0.7696	0.7373	0.6996

Table 4: The results of PRDetect in short text detection experiments.

The detection of short texts poses significant challenges for LLM text detectors (McGovern et al., 2024).

As shown in Table 4, the performance of PRDetect gradually decreases with the reduction in length. This is because the average length of the texts in the GPT3.5-Mixed dataset is quite long, making the decrease in performance on short texts more pronounced.

C Text analysis in the dataset

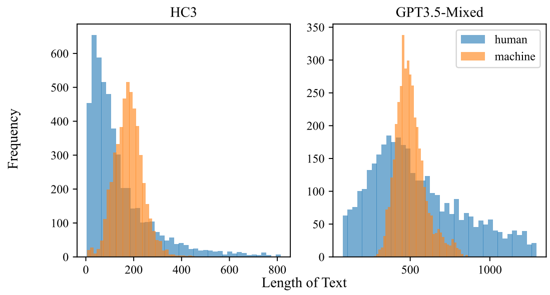

Figure 3: The length distribution of the dataset. To facilitate presentation, some excessively long instances were excluded when creating the graph.

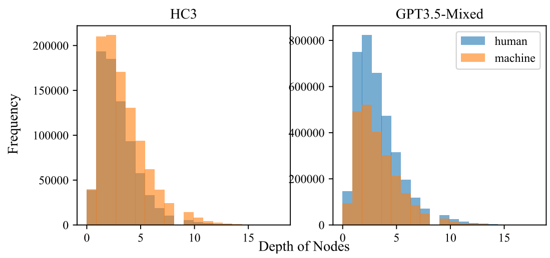

Figure 4: The average node depth in the syntactic trees of the dataset.

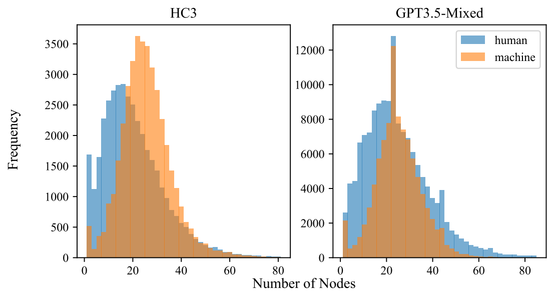

Figure 5: The average number of nodes in the syntactic trees of the dataset.

Figure 3 demonstrates the difference in length between human-written and machine-generated texts in the two datasets. Figure 4, 5, 6 demonstrate the differences in syntax trees between human-written and machine-generated texts in the datasets. The distribution differences in syntax trees determine the effectiveness of the methodology employed in this experiment.

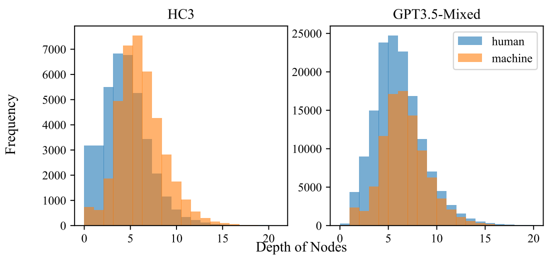

Figure 6: The average height of root nodes in the syntactic trees of the dataset.