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

Xiang Li[αβ] **Zhiyi Yin**[α] **Hexiang Tan**[αβ] **Shaoling Jing**[α]
Du Su[α] **Yi Cheng**[α] **Huawei Shen**[αβ] **Fei Sun**[α]
[α]CAS Key Laboratory of AI Safety, Institute of Computing Technology, CAS
[β]University of Chinese Academy of Sciences
{lixiang22s, tanhexiang21s, jingshaoling, shenhuawei, sunfei}@ict.ac.cn

Abstract

As LLM-generated text becomes increasingly prevalent on the internet, often containing 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 introduce perturbations to the LLM-generated text, and the robustness of existing detection methods against these perturbations has not been sufficiently explored. This paper empirically investigates this challenge and finds that even minor perturbations 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 distinct differences between human-written and LLM-generated text. Therefore, we propose a detection method based on syntactic trees, which can capture features invariant to perturbations. It demonstrates significantly improved robustness against perturbation on the HC3 and GPT-3.5-mixed datasets. Moreover, it also has the shortest time expenditure. We provide the code and data at `https://github.com/thulx18/PRDetect`.

1 Introduction

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

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

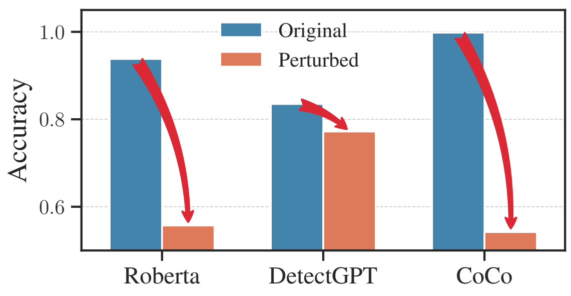

Figure 1: The accuracy of several detection methods drop significantly after perturbing just 10% words of each LLM-generated sentence in the HC3 datasets.

Recently, some studies have begun to define perturbations and explore how to accurately identify text after perturbation. Perturbations can be broadly categorized into sentence-level, token-level, and character-level disturbances (Zhou et al., 2024). These perturbation methods reduce the accuracy of existing classifiers (Liu et al., 2023b, 2024; Huang et al., 2024). Furthermore, some studies have attempted new methods to enhance robustness (Zhang et al., 2024). Zhou et al. (2024) argues that traditional text classification methods heavily rely on statistical metrics, such as perplexity. Perturbations in the text lead to changes in text quality and readability, among other aspects, resulting in decreased classification performance. Liu et al. (2023b) proposes a text detection method based on an entity graph, which shows some resistance to token-level perturbations, illustrating the effectiveness of graph information in improving robustness to perturbations.

To address this problem, we identify differences in the syntax trees between human-written texts and LLM-generated texts, as 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. We compare PRDetect with several highperforming baselines under perturbation. PRDetect demonstrates both high accuracy and perturbation-robustness. Additionally, we discuss the performance of PRDetect in other contexts, including cross-dataset and short-text experiments. The results indicate that as text length decreases and the perturbation ratio increases, classification accuracy declines.

In summary, our contributions are as follows:

- PRDetect leverages 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 exhibits state-of-the-art robustness against perturbation.

2 Related Work

2.1 LLM-Generated Text Detection

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

Featured-based text detection: 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) utilizes an open-source language model to create a local probability dictionary and calculate perplexity for classification, which helps save storage space. CoCo (Liu et al., 2023b) leverages linguistic features within the text, representing the entity information as a graph structure, which is subjected to contrastive learning. It performs well in detecting long texts.

Fine-tuning large pre-trained model: Pre-trained language models offer significant advantages in NLP tasks, eliminating the need for manually specified features. Transformer-based models can extract useful features from text for classification tasks, such as determining whether a piece of text was generated by ChatGPT or written manually (Mitrović et al., 2023). OpenAI fine-tuned a RoBERTa model[1] to detect text generated by GPT-2. Their dataset comprises 250,000 documents from the WebText test set, along with 500,000 text samples from GPT-2 models of varying parameter sizes. These methods can be further refined and enhanced by fine-tuning with local data.

Zero-shot method: This approach relies on certain statistical regularities, saving time in model training and representing a significant breakthrough in the task of LLM-generated text detection. DetectGPT (Mitchell et al., 2023) finds that machine-generated text tends to occupy regions of negative curvature in the model's log-probability function. By perturbing the text and calculating changes in log probability, texts with smaller average changes are more likely to be human-written. DetectGPT-SC (Wang et al., 2023), based on masked prediction consistency, also achieves zero-shot classification. Fast-DetectGPT (Bao et al., 2024) optimizes the sampling stride, accelerating the detection process up to 340 times. DetectGPT4Code (Yang et al., 2023) achieved state-of-the-art results by experimenting using the CodeContest and APPS datasets.

Text watermarking method: Adding a watermark involves embedding a hidden representation into the text, making it indistinguishable to humans but detectable by algorithms. In simple terms, adding a watermark incorporates a selection strategy into the text generation process. When prior tokens are available, a random seed is generated by computing a hash on the last token before before generating the next word. This seed is then used to divide the vocabulary into a green list and a red list, with the next word being selected only from the green list (Kirchenbauer et al., 2023). Generally, watermarks only need to be added during generation without the need to retrain the model. The watermarks should be difficult to remove or modify and detectable even in partial text. To ensure the watermark's effectiveness, it is crucial to use methods that do not leak. Measures must be taken to prevent the watermark from being erased or counterfeited, such as generating a key to produce random numbers used to create a private watermark (Kirchenbauer et al., 2023). Another method is to use two separate neural networks for watermark generation and detection (Liu et al., 2023a).

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

2.2 Text Perturbation

Existing experiments have demonstrated that simple perturbations can significantly interfere with popular text detectors. These perturbations typically alter the underlying distribution of the text, resulting in incorrect classifications.

One type of character-level perturbation can be implemented by replacing characters with visually similar letters from different languages (Wolff and Wolff, 2020). While these may be indistinguishable to the human eye, they are represented as different tokens by the classifier. Another approach mimics human spelling errors by swapping letters within words. Adversarial perturbation experiments often involve character-level substitutions, deletions, and insertions (Huang et al., 2024). Additionally, there are character-level perturbations that can affect detection results, such as word merging, capitalization errors, punctuation removal, and space insertion (Zhou et al., 2024).

Some studies have conducted token-level perturbation experiments on text (Liu et al., 2023b), including the random insertion, substitution, deletion, and repetition of tokens. However, such perturbations typically do not occur in real-world scenarios and can also undermine the readability of the text.

Additionally, some studies assess sentence-level modifications (Macko et al., 2024). One approach involves translating text from one language to another and then back-translating it to the original language. Through back-translation, discrepancies can arise between the text translated and the original. Furthermore, the text can be rewritten or subjected to various text attacks. These methods can evade detection by some classifiers.

In experiments, character-level and word-level modifications can be more effectively quantified and analyzed. Sentence-level alterations, however, are challenging to control quantitatively in terms of the degree of perturbation. Therefore, in the robustness analyses of many studies, only simple text perturbations, such as deletions, insertions, substitutions, and repetitions are typically conducted.

3 Methods

The primary framework of PRDetect consists of constructing syntax trees, node encoding, supervised training of a graph convolutional network, and applying text perturbations for testing purposes. The main process is illustrated in Figure 2.

3.1 Syntax Tree Construction and Node Encoding

Stanford University has conducted extensive research on the construction of syntax trees (Berant et al., 2013). Stanford CoreNLP[2] is a suite of text analysis tools that supports multiple languages. It can determine the part of speech for words in the text and identify over 50 types of grammatical dependency relationships. SpaCy[3] is another open-source software library for NLP, developed in Python and Cython, and designed specifically for production use. It offers a wide range of linguistic annotations and features, such as part-of-speech tagging, parsing, named entity recognition, and more.

In this paper, we utilize SpaCy and Roberta to accomplish this process. For a given long input text, we use SpaCy for tokenization after segmenting the text into chunks. SpaCy performs part-of-speech tagging on each token and determines dependency relationships using a set of rules, such as identifying the subject-verb or modifier relationships. Through this process, a dependency tree is constructed, where each node represents a token. Subsequently, we construct an adjacency matrix A, where 1 represents a dependency relationship between two tokens, and 0 otherwise, based on the dependency tree. The adjacency matrix of a syntax tree is represented as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } s_i.head == s_j \\ 0 & \text{others} \end{cases} \tag{1}$$

where s_i denotes the i-th word of the text T. These operations are completed during the text preprocessing stage. Given the sparsity of the syntax tree's adjacency matrix, a sparse matrix is stored to save space.

For the token nodes of the dependency tree, we use Roberta to obtain their embedding, which are used to initialize the nodes in the graph network (Liu et al., 2023b). Compared to random initialization, this approach leads to faster convergence and improved performance.

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

[2]https://github.com/stanfordnlp/CoreNLP
[3]https://github.com/explosion/spaCy

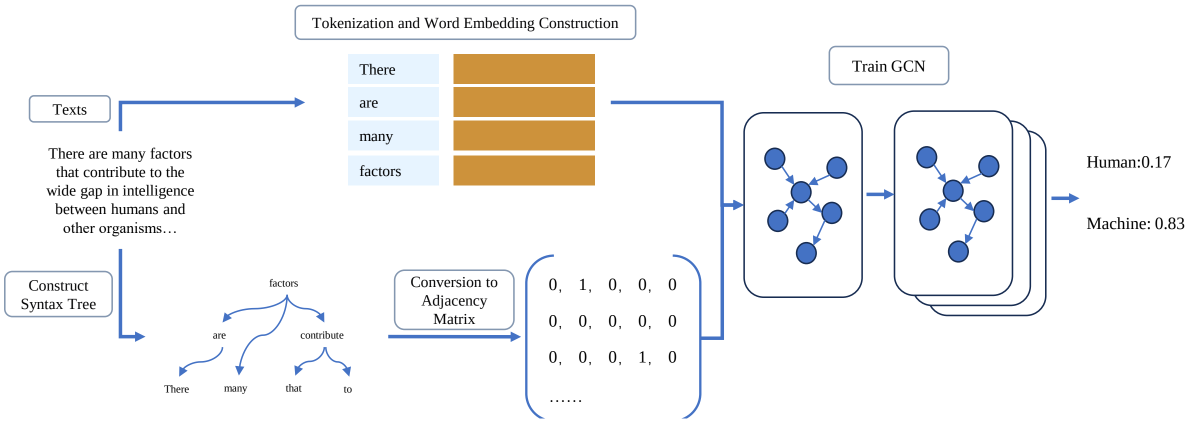
Tokenization and Word Embedding Construction
There
are
many
factors
Train GCN
Texts
There are many factors that contribute to the wide gap in intelligence between humans and other organisms…
Human:0.17
Machine: 0.83
Construct Syntax Tree
factors
are
contribute
There
many
that
to
Conversion to Adjacency Matrix
0, 1, 0, 0, 0
0, 0, 0, 0, 0
0, 0, 0, 1, 0
……

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

3.2 Graph Convolutional Network

Graph Convolutional Network (GCN) (Kipf and Welling, 2017) is a deep learning model specifically designed for processing data with graph structures. It learns embedded representations of nodes by performing convolutional operations on the graph, effectively capture the local connection patterns between nodes. By stacking multiple convolutional layers, GCN achieves hierarchical abstraction of node features and enables in-depth exploration of information. Due to its excellent performance and flexibility, GCN has been widely applied across various domains, and has achieved promising results in numerous classification tasks involving graph-structured data.

In this paper, we utilize two layers of GCN to perform graph convolution operations. Each convolution layer can be expressed as:

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

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 nonlinear activation function, typically the Rectified Linear Unit (ReLU), which outputs the input directly if it's positive and zero otherwise. Self-loops for nodes can reinforce the inherent features of the nodes during the convolution process, represented as:

$$\hat{A} = A + I \quad (3)$$

where I is the identity matrix of the same dimension as A. Our model employs the Binary Cross-Entropy Loss (BCELoss) as the loss function L, represented as:

$$L = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})) \quad (4)$$

where y is the true binary label of the sample, with 1 indicating a human-written text and 0 indicating a machine-generated text. The $\hat{y}$ is the predicted probability output by the model.

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.

4 Dataset and Syntactic Tree Difference Analysis

In this section, we introduce the datasets and metrics used in our experiments. We also analyze the human-written and LLM-generated texts within the dataset. The differences in syntax trees features are the decisive factors in classification.

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 harder for humans to distinguish from human-written text.

Human ChatGPT Comparison Corpus

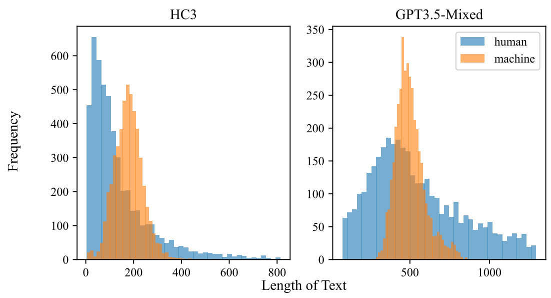

Figure 3: The length distribution of the dataset. To facilitate presentation, some excessively long instances are excluded.

(HC3)[4] (Guo et al., 2023): This dataset contains questions and answers from both ChatGPT and human experts, including domain-specific experts, high-voted answers by web users, Wikipedia, and Baidu Baike. The dataset includes both Chinese and English text and covers domains such as open-domain topics, computer science, finance, medicine, law, and psychology. We exclude invalid samples, such as instances where ChatGPT declines to provide an answer.

GPT3.5-Mixed[5] (Liu et al., 2023b): This dataset is generated by text-davinci-003 and focuses 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, Yahoo, CNBC and Times.

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

4.2 Syntactic Tree Difference Analysis

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

Table 1 presents the average number of nodes in the syntax tree, the average height of the root node, and the average depth of nodes per tree. 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. These differences enable the GCN to learn and classify the texts correctly.

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

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

Fig 3 presents the average length of human-written and LLM-generated texts in the dataset. The difference in length between the two datasets also has a certain impact in the experiments, shown in the Subsecion 5.6. Detailed analysis and distribution graphs can be found in the Appendix A.

5 Experiments

In this section, we first introduce the baselines used for comparison. Next, we describe our main perturbation methods. We then compare the classification accuracy of verious methods on both LLM-generated texts and perturbed texts to demonstrate the state-of-the-art performance of PRDetect. Finally, we conduct analyses on cross-dataset performance, the effects of other perturbation types, experiments with short texts, and classification efficiency.

5.1 Baselines

In our study, we compared PRDetect with several commonly used or state-of-the-art detectors designed for LLM-generated text identification. Text watermarking is not included, as it typically requires intervention prior to text generation, and its experimental setup differs from the other methods described in Subsection 2.1.

RoBERTa (Liu, 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[6].

DetectGPT (Mitchell et al., 2023) is a zero-shot LLM-text detection method that distinguishes texts by examining changes in the log-probability curves under perturbation. In the experiments, the T5-large model is used to perturb texts, while the GPT-medium model serves as the scoring model.

CoCo (Liu et al., 2023b) leverages entity information to train a text detection model. An entity graph is constructed based on different entities within the same sentence or the same entities across different sentences.

5.2 Text Perturbation in Experiments

In this paper, our primary experiments focus on word-level synonym replacement. This method allows us to perturb the text while maintaining its contextual meaning as much as possible. During the construction of the syntax tree in subsection

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

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	0.9878 ±0.0010	**0.9878 ±0.0010**	**0.9872 ±0.0009**	**0.9874 ±0.0007**	**0.9864 ±0.0005**	0.9656 ±0.0007	**0.9630 ±0.0004**	**0.9632 ±0.0006**	**0.9656 ±0.0010**	**0.9638 ±0.0011**

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. To demonstrate statistical significance, we selecte 5 different seeds. In this table, mean and standard deviation of PRDetect are reported.

3.1, we obtain the part of speech for each word and select a category of words for marking. We then use WordNet, a component of the NLTK[7] toolkit, to obtain a list of synonyms for each word. From this list, we choose to replace the original word with the first synonym listed. There are also various other methods for selecting replacement words, and some LLMs can be utilized for synonym replacement.

When replacing synonyms for adjectives, we select proportions of 5%, 10%, 20%, and 30%. It is important to avoid modifying more than 50% off the original text, as this makes it challenging to define the label of the perturbed text. The perturbed texts obtained by this way are used exclusively in the model's testing phase. Additionally, this paper compares other word-level perturbations in Subsection 5.5.

5.3 Main Experiments

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

5.3.1 Parameter Settings

In this paper, we utilize two-layer $GCNConv$ to perform graph convolution operations, followed by a fully connected layer and a dropout layer. The dropout layer is set with a parameter of 0.5. In the main experiments, we use random seeds ranging from 2021 to 2025. The GPU we utilized is the RTX 4090.

5.3.2 Detecting LLM-generated texts

. We train both CoCo and PRDetect using the same training set and subsequently tested all methods on the same test set. 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. It meets the requirements for practical application and achieves an optimal balance between accuracy and robustness. Both PRDetect and CoCo, which utilized GCN to learn graph features, outperformed the other two methods based on semantic features, proving the effectiveness of graph information in text detection.

RoBERTa, which was trained on texts from GPT-2, showed strong performance on texts from ChatGPT and GPT-3.5, highlighting its robust generalization capabilities. However, its accuracy is not as high as PRDetect's.

DetectGPT, serving as a zero-shot classifier, achieves optimal results when the scoring model is consistent with the generation model–a condition difficult to meet in practical detection scenarios, which highlighte its limitations. Additionally, DetectGPT struggles with detecting long texts, which is an issue noted on its official Github[8].

5.3.3 Detecting Perturbed Texts

We perturd the test set texts according to the method described in Subsection 5.2. By selecting proportions of adjectives at 5%, 10%, 20%, and 30%, we generate the perturbed texts. These perturbed texts are then used to test the four methods under the same conditions.

Table 2 shows 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 greater decrease in accuracy as the intensity of perturbations increases.

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

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

In contrast, the other baselines experience a decrease in accuracy as perturbation intensity increases. RoBERTa tends to predict perturbed texts as human-written, resulting in a detection accuracy rate of around 50% for perturbed texts. Even with minimal perturbation of just 5%, its accuracy suffers a catastrophic decline. DetectGPT, due to its inherent perturbation process within its detection workflow, exhibits some degree of resilience against perturbations. On the HC3 dataset, with a 5% perturbation, its accuracy declines by approximately 3%, demonstrating the robustness characteristic of zero-shot methods. However, DetectGPT's effectiveness significantly diminishes when handling longer texts. On the GPT3.5-Mixed dataset, its accuracy rate is marginally above 50%. CoCo, while highly accurate on original texts, experiences a catastrophic drop in detection accuracy on perturbed texts. Nevertheless, its performance on the GPT3.5-Mixed dataset is better than on the HC3 dataset, highlighting the effectiveness of entity graph features in longer texts.

In summary, PRDetect demonstrates a strong ability to resist text perturbation while maintaining a high detection accuracy rate. In Section 5.5, we will further compare these methods with other perturbations, showcasing PRDetect's perturbation robustness.

5.4 Cross-Dataset Experiments

We conducted cross-experiments on the HC3 and GPT3.5-Mixed datasets, with the results presented in Table 3. These two datasets differ in terms of text length and domain. The texts in the GPT3.5-Mixed dataset are significantly longer than those in the HC3 dataset, as can be seen in Figure 3. HC3 covers various fields, such as economics, law, and medicine, while GPT3.5-Mixed pertains to the news domain.

It can be observed from the table that PRDetect demonstrates a notable level of accuracy and perturbation resistance in cross-dataset experiments. Furthermore, the classification accuracy of the GCN trained on longer texts is superior. A more detailed discussion regarding the impact of varying text lengths is provided in Subsection 5.6.

The detection of LLM-generated text using parse trees is influenced by the text's domain. The HC3 dataset consists of responses from humans and ChatGPT to identical questions, whereas the GPT3.5-Mixed dataset is a collection of generated news articles. The latter follows the fundamental structure of news, including elements such as headlines, leads, main topics, and conclusions. These disparities in text domains lead to variations in syntactic style, which in turn result in diminished performance for PRDetect in cross-dataset experiments.

Train	HC3	GPT3.5-Mixed
Test	GPT3.5-Mixed	HC3
0%	73.2%	87.7%
5%	73.0%	87.8%
10%	72.6%	87.3%
20%	71.7%	86.9%
30%	70.9%	87.0%

Table 3: Accuracy of PRDetect in cross-dataset experiments.

Model	CoCo	PRDetect
Original	0.9981	0.9850
Insert	0.4733	**0.9830**
Repeat	0.5380	**0.9820**
Replace	0.4713	**0.7980**
Detect	0.5212	**0.7470**
Average	0.5010	**0.8775**

Table 4: Accuracy on four common types of perturbation. The experiments are conducted on the HC3 dataset. In this experiment, the seed is set to 2024.

5.5 Other Perturbation Experiments

In some papers (Liu et al., 2023b, 2024), the authors randomly **insert**, **delete**, **repeat** or **replace** words to perturb the text. We apply these four types of perturbations at a 25% ratio and detect on the HC3 test dataset.

These types of perturbations significantly affect certain classifiers, as random operations can impact the quality and readability of the text, leading to noticeable changes in perplexity (Zhou et al., 2024). Consequently, the statistical features learned by these classifiers become ineffective.

As shown in Table 4, PRDetect outperforms CoCo, another detection method that leverages graph information, on perturbed texts. PRDetect is minimally affected by Insert and Repeat perturbations, as these modifications have little impact on the original syntax tree. For the other two perturbation types, which alter the syntax tree, the detection

Length	Acc	F1
Original	0.9610	0.9617
$[80, 100)$	0.6750	0.7257
$[60, 80)$	0.6700	0.7179
$[40, 60)$	0.6900	0.7257
$[20, 40)$	0.6850	0.7273
$[10, 20)$	0.5850	0.6770

Table 5: The results of PRDetect in short text detection experiments, which are conducted on the GPT3.5-Mixed dataset. "Length" refers to the number of tokens in the text after being split by spaces.

accuracy of PRDetect declines but still maintains strong performance.

5.6 Short Text Detection

Detecting short texts poses significant challenges for LLM text detectors (McGovern et al., 2024), as they often rely on contextual cues that may not be present in brief passages.

In this experiment, we segment and sample the test data from the GPT3.5-Mixed dataset based on varying lengths. Testing is conducted using the model preserved from the main experiment. We compare the detection results of different text lengths.

As shown in Table 5, there is a clear trend where the performance of the PRDetect model gradually diminishes as text length decreases. This observed decline in performance on shorter texts is attributed to the predominance of longer texts in the GPT-3.5-Mixed dataset, leading to a more significant impact on the model's performance when handling with shorter text segments. Consequently, PRDetect appears to be less proficient in detecting shorter text segments, which is a critical aspect to consider for further refinement of the detection algorithm.

5.7 Efficiency

In real-world scenarios, online texts are being generated every minute. This requires detectors to have sufficiently high efficiency.

In this experiment, we compared the time spent by PRDetect and other baselines at each stage, with the results shown in Table 6. We control the equipment, training epochs, and data to be exactly the same in our experiments. Then, we record the time spent by each method at every stage.

The results indicate that PRDetect is faster than other baselines at every stage. Although DetectGPT has shown certain perturbation-robustness on HC3 shown in Table 2, it spends an extremely high amount of time during detection. Compared to CoCo, PRDetect reduces the test time spent to one-tenth, which is highly significant in practical applications because the texts to be detected are often voluminous. Detection time is an indispensable factor in evaluating detection tools. However, since Coco and PRDetect utilize graph information within the text, a substantial amount of time is spent in the preprocessing stage converting text into adjacency matrices, which is an area for potential future improvement.

Model	Preprocessing (s)	Train (s)	Test (s)
RoBERTa	-	-	27.6
DetectGPT	-	-	1298.4
CoCo	2545.3	1655.0	28.3
PRDetect	**1754.5**	**629.7**	**2.5**

Table 6: The time consumed by PRDetect and other baselines at each stage. "Preprocessing" refers to the time required for graph construction from the text. "Train" refers to the time required for training the model. Among them, RoBERTa is a pre-trained model, and DetectGPT is a zero-shot method, so their training times are not considered. The number of training epochs is fixed at 15. "Test" refers to the duration required for testing the model. The experiments are conducted on an RTX 4090. The dataset utilized is GPT3.5-Mixed.

6 Conclusion

In this paper, we propose PRDetect, a perturbation-robust detection method for LLM-generated text that leverages differences in syntax trees to train a GCN. It identifies generated text effectively and shows strong perturbation robustness. To simulate the polishing of generated text before its actual use, we introduce a perturbation method based on synonym replacement, It perturbs the text while maintaining its readability. PRDetect is minimally affected by text perturbations on the HC3 and GPT3.5 datasets, and its accuracy is significantly higher than that of other baselines. Additionally, PRDetect spends the least amount of time during training and testing, which demonstrates its superiority in practical applications, particularly valuable in scenarios where rapid detection is necessary, such as in content moderation or academic integrity assessments.

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 at the token-level. We have not tested methods such as backtranslation and rewriting at the sentence-level for two reasons. First, sentence-level perturbations significantly impact the graph structure, making detection 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 more than 50% perturbation 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 Subsection 5.6 and 5.4, the performance of PRDetect, when trained on long texts, significantly declines when the text length falls below 100 tokens, with accuracy levels between 0.58 and 0.68. 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 0.88 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.

Acknowledgements

This work was supported by the National Key R&D Program of China (2022YFB3103700, 2022YFB3103704), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDB0680202), and the Innovation Funding of ICT, CAS (E361120).

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. *ICLR 2024*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

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, SEAS Harvard, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *ACL 2019*, page 111.

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. *arXiv 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? *ACL 2024*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR 2017*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and S Yu Philip. 2023a. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Shengchao Liu, Xiaoming Liu, Yichen Wang, Zehua Cheng, Chengzhengxu Li, Zhaohan Zhang, Yu Lan, and Chao Shen. 2024. Does detectgpt fully utilize perturbation? bridging 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. 2023b. Coco: Coherence-enhanced machine-generated text detection under low resource with contrastive learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16167–16188.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv 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. *EMNLP 2024 Findings*.

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. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

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. 2020. Attacking neural text detectors. *ICLR 2020 workshop "Towards Trustworthy ML: Rethinking Security and Privacy for ML."*.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. Llmdet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2113–2133.

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. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Yi-Fan Zhang, Zhang Zhang, Liang Wang, and Rong Jin. 2024. Assaying on the robustness of zero-shot machine-generated text detectors. *AAAI 2024 Workshop on Responsible Language Models*.

Ying Zhou, Ben He, and Le Sun. 2024. Navigating the shadows: Unveiling effective disturbances for modern ai content detectors. *ACL 2024*.

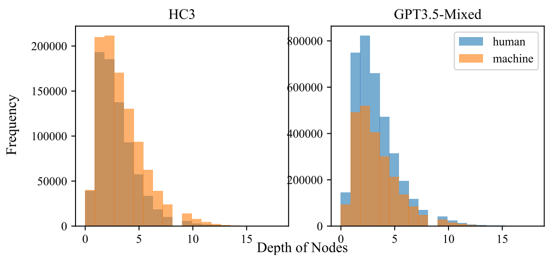

Figure 4: The average node depth in the syntactic trees of the dataset.

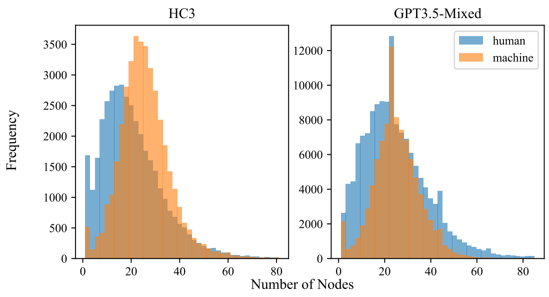

Figure 5: The average number of nodes in the syntactic trees of the dataset.

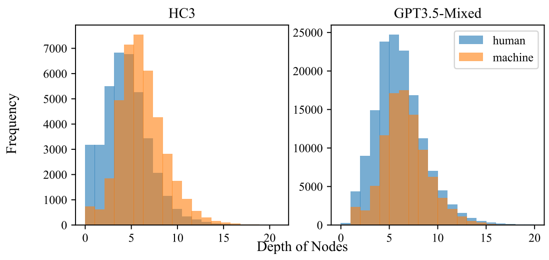

Figure 6: The average height of root nodes in the syntactic trees of the dataset.

A Text Analysis in the Dataset

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.

B The Impact of Perturbations on Syntax Trees

In Table 7, we show the changes in syntactic tree features under different perturbation ratios.

Labels	Perturbation Ratio(%)	Depth of Nodes	Number of Nodes	Height of Root
ChatGPT	0	3.2620	25.1101	6.3754
ChatGPT	5	3.2266	24.9980	6.2874
ChatGPT	30	3.2245	25.0424	6.2913
Human	0	2.7268	18.9947	4.5711
Human	5	2.7172	18.9029	4.5480
Human	30	2.7201	18.8982	4.5476

Table 7: Analysis of the changes in the syntax tree structure of test samples before and after perturbation in the HC3 dataset

Labels	0%	5%	30%
HC3	0.9338	0.9190	0.9022
GPT3.5-Mixed	0.9540	0.9550	0.9560

Table 8: Under the similar text distribution, the recognition accuracy of PRDetect.

Label	0%	30%	Decline Ratio
HC3	82.31	75.50	8.27%
GPT3.5-Mixed	85.20	77.95	8.51%

Table 9: The readability scores assigned by ChatGPT-4o mini and the percentage decrease in scores before and after perturbation.

Table 7 can illustrate two points:

- The syntax tree features of humans and machines are different, which is the key to classification.
- Token-level perturbations have a minimal impact on syntax tree features, which is the key to robustness against perturbations.

C Detection under the Similar Length Distribution

In this paper, we utilized datasets from other studies. Among them, there are differences in the length distribution between texts generated by LLMs and those written by humans.

To address this difference, we select pairs of texts with similar lengths to reduce the differences in length distribution. Then, we use the previously trained model for detection. The accuracy is as follows in the table below. As shown in Table 8, the accuracy of PRDetect remains above 90%.

D Readability

Table 10 shows a few examples from test samples with the largest replacement ratios. To analyze the impact of perturbation on sentence readability, we used ChatGPT-4o mini to assess the readability of the text. The prompt used was: "Whether the following text is valid for the human reader? Ranking it from 0 to 100. Only return the score:text". The results are presented in Table 9. The score has decreased by approximately 8.5%.

Original text	Perturbed text
Do you use any other *online* features of Quicken? How many *unique* ticker symbols do you have?	Do you use any other *on-line* features of Quicken? How many *singular* ticker symbols do you have?
Ending up with nothing is an *unlikely* situation unless you invest 100% in a company stock and the company goes under.	Ending up with nothing is an *improbable* situation unless you invest 100% in a company stock and the company goes under.
The electoral college almost always votes the way the *popular* vote does.	The electoral college almost always votes the way the *pop* vote does.

Table 10: The changes in the text before and after perturbation. The changed parts are indicated in italic.