Zero-Shot Detection of LLM-Generated Text using Token Cohesiveness

Shixuan Ma and **Quan Wang**[*]
MOE Key Laboratory of Trustworthy Distributed Computing and Service,
Beijing University of Posts and Telecommunications
{1132685456, wangquan}@bupt.edu.cn

Abstract

The increasing capability and widespread usage of large language models (LLMs) highlight the desirability of automatic detection of LLM-generated text. Zero-shot detectors, due to their training-free nature, have received considerable attention and notable success. In this paper, we identify a new feature, token cohesiveness, that is useful for zero-shot detection, and we demonstrate that LLM-generated text tends to exhibit higher token cohesiveness than human-written text. Based on this observation, we devise TOCSIN, a generic dual-channel detection paradigm that uses token cohesiveness as a plug-and-play module to improve existing zero-shot detectors. To calculate token cohesiveness, TOCSIN only requires a few rounds of random token deletion and semantic difference measurement, making it particularly suitable for a practical black-box setting where the source model used for generation is not accessible. Extensive experiments with four state-of-the-art base detectors on various datasets, source models, and evaluation settings demonstrate the effectiveness and generality of the proposed approach. Code available at: https://github.com/Shixuan-Ma/TOCSIN.

1 Introduction

The past few years have witnessed tremendous advances in Large Language Models (LLMs). These models such as ChatGPT (OpenAI, 2022), PaLM (Chowdhery et al., 2023), GPT-4 (OpenAI, 2023) can now generate text of supreme quality, demonstrating exceptional performance in various fields like question answering, news reporting, and story writing. The increasing capability of LLMs to produce human-like text at high efficiency, however, also raises concerns about their misuse for malicious purposes, e.g., phishing (Panda et al., 2024), disinformation (Jiang et al., 2024), and academic dishonesty (Perkins, 2023). The effective detection of LLM-generated text therefore becomes a vital principle to ensure the responsible use of LLMs.

[*]Corresponding author: Quan Wang.

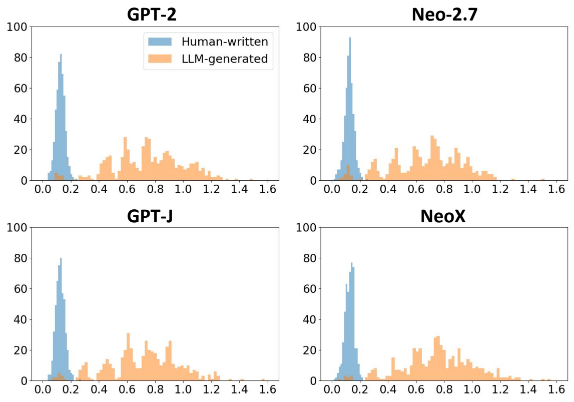

Figure 1: Histograms of token cohesiveness distributions for 500 human-written and 500 LLM-generated articles. Human-written articles are sampled from XSum (Narayan et al., 2018), and LLM-generated articles are produced by prompting four source models with the first 30 tokens of each human-written article. The calculation of token cohesiveness will be detailed in Section 3.2.

LLM-generated text detection is typically formulated as a binary classification task, i.e., to classify if a piece of text is generated by a particular source LLM or written by human (Tang et al., 2024). Current solutions roughly fall into two categories: supervised classifiers and zero-shot classifiers. Supervised classifiers are trained from labeled data and thus may overfit to their specific training domains (Wang et al., 2023b). Zero-shot classifiers, in contrast, are entirely training-free, making them less prone to domain-specific degradation and typically generalizing better (Zhu et al., 2023).

Most existing zero-shot detectors are developed based on the generation probabilities (or their variations) of the source model, assuming that LLM-generated text aligns better with these probabilities (Gehrmann et al., 2019; Mitchell et al., 2023). This paper takes a different tack and introduces a fundamentally new feature, **token cohesiveness**, which

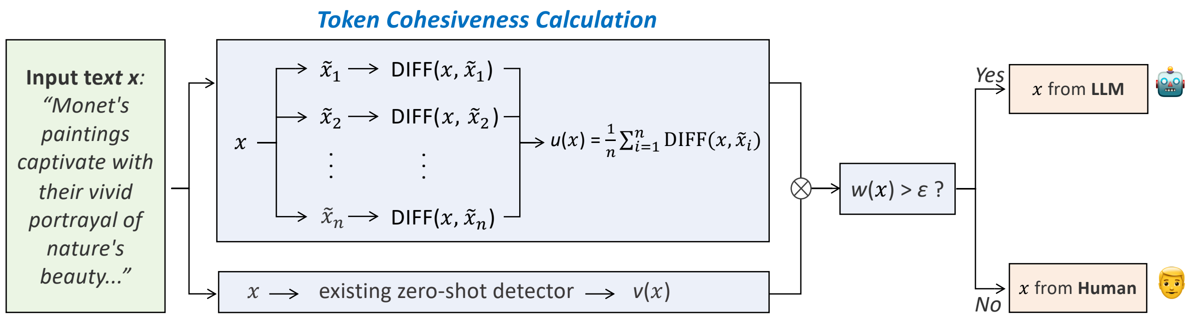
Token Cohesiveness Calculation
Input text x: "Monet's paintings captivate with their vivid portrayal of nature's beauty..."
x
$\tilde{x}_1 \rightarrow \mathrm{DIFF}(x, \tilde{x}_1)$
$\tilde{x}_2 \rightarrow \mathrm{DIFF}(x, \tilde{x}_2)$
$\tilde{x}_n \rightarrow \mathrm{DIFF}(x, \tilde{x}_n)$
$u(x) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{DIFF}(x, \tilde{x}_i)$
$x \rightarrow$ existing zero-shot detector $\rightarrow v(x)$
$w(x) > \varepsilon$?
Yes
x from LLM
No
x from Human

Figure 2: Overview of TOCSIN. The input text x is fed into the upper channel to calculate token cohesiveness $u(x)$, and the lower channel to produce raw prediction $v(x)$. The two scores are then combined into $w(x)$, and if the combination exceeds a predefined threshold ϵ, the text x is categorized as LLM-generated.

does not rely on the source model's output to detect LLM-generated text. Token cohesiveness is defined as the expected semantic difference between input text x and its copy $\tilde{x}$ after randomly removing a small proportion of tokens. It essentially measures how closely the removed tokens are semantically related to the rest of the input text. Then our key assertion is that **LLM-generated text generally exhibits higher token cohesiveness than human-written text**. This is because LLMs use the causal self-attention mechanism (Vaswani et al., 2017) to generate text, requiring the generation of each token to be conditioned on all its preceding tokens, which would naturally foster a closer relationship among tokens, thus increasing token cohesiveness. In contrast, humans tend to write text more freely with no such explicit restriction, which potentially results in a looser relationship among tokens, thus reducing token cohesiveness. We empirically verify this assertion, and find that it holds true across a diverse body of LLMs, as illustrated in Figure 1.

Given the discriminative power of the new feature and its distinctiveness from existing detectors, we propose **TOCSIN**, a novel paradigm that leverages TOken CoheSIveNess to enhance zero-shot detection of LLM-generated text. TOCSIN is a dual-channel detector, with one channel equipped with an existing zero-shot detector, and the other channel a token cohesiveness calculation module. Given a piece of text to be detected, we create multiple copies, with each copy randomly removing a specific proportion of tokens from the input text. We calculate the average semantic difference, or more precisely, the average negative BARTScore (Yuan et al., 2021) between the input text and these copies as the token cohesiveness score. Meanwhile, we feed the input text into an existing zero-shot detector to produce a raw prediction score. After that, we combine the two scores and perform thresholding on the combination to make the final decision. The overall procedure is sketched in Figure 2.

As an enhancement to existing zero-shot detectors, TOCSIN enjoys several merits. (1) **General applicability:** The dual-channel solution of TOCSIN is quite generic, allowing token cohesiveness to be used as a plug-and-play module in a variety of detectors. (2) **Low additional time cost:** TOCSIN keeps the existing detector channel unchanged, and the additional time cost mainly comes from token cohesiveness calculation, which only involves several rounds of random token deletion and BARTScore computation, and is highly time efficient. (3) **Low additional space cost:** The only additional space cost comes from loading BART (Lewis et al., 2020) (or more precisely, BART-base), which is relatively small compared to those scoring models used in existing zero-shot detectors.

To rigorously evaluate the effectiveness and generality of TOCSIN, we apply it to four current state-of-the-art zero-shot detectors, Likelihood (Mitchell et al., 2023), LogRank (Mitchell et al., 2023), LRR (Su et al., 2023) and Fast-DetectGPT (Bao et al., 2024), and conduct extensive experiments on four diversified datasets, with LLM-generated passages produced by eight different source models ranging from GPT-2 (Radford et al., 2019) to GPT-4 (OpenAI, 2023) and Gemini (Google, 2023). Experimental results demonstrate consistent and meaningful improvements over the four detectors in both white-box and black-box settings.

Our main contributions in this paper are threefold: (1) unveiling and validating a new hypothesis that LLM-generated text exhibits higher token cohesiveness than human-written text, (2) proposing a novel and generic framework that uses token cohesiveness to improve zero-shot detection of LLM-generated text, and (3) achieving new best detection accuracy compared to existing zero-shot detectors.

2 Related Work

The increasingly powerful LLMs (Radford et al., 2019; OpenAI, 2022, 2023; Chowdhery et al., 2023; Touvron et al., 2023), though demonstrating excellent performance on various language-related tasks, raise numerous ethical concerns, drawing extensive attention to automatic detection of LLM-generated text (Guo et al., 2023; Li et al., 2023b).

LLM-generated text detection is typically formulated as a binary classification task, with current solutions roughly categorized into supervised classifiers and zero-shot classifiers. Supervised classifiers are those trained with statistical features (Solaiman et al., 2019; Ippolito et al., 2020; Wu et al., 2023; Verma et al., 2023) or neural representations (Uchendu et al., 2020; Bakhtin et al., 2019; Zhong et al., 2020; Bhattacharjee et al., 2023; Wang et al., 2024) to discriminate LLM-generated and human-written text, wherein a popular trend is to directly fine-tune a pre-trained language model like BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019) for the classification task (Zellers et al., 2019; Rodriguez et al., 2022; Mitrović et al., 2023; Chen et al., 2023). These supervised classifiers, though achieving excellent performance on their training domains, require periodic retraining to adapt to new LLMs, and often exhibit a tendency to overfit their training data (Pu et al., 2023).

Zero-shot classifiers are entirely training-free and often show better generalization ability. Their key ideas are to extract various statistical features, e.g., likelihood (Hashimoto et al., 2019), perplexity (Lavergne et al., 2008), normalized log-rank perturbation (Su et al., 2023), probability and conditional probability curvatures (Mitchell et al., 2023; Bao et al., 2024), and perform thresholding on these features to discern LLM-generated and human-written text. Such features can be collected from the source LLMs themselves (the white-box setting) or from surrogate models (the black-box setting). In this paper, we propose a new feature called token cohesiveness to improve existing zero-shot detectors. The computation of the new feature does not rely on the source LLMs, making it particularly suitable for the black-box setting.

Recently there emerge some black-box zero-shot detectors based on LLM rewriting (Zhu et al., 2023; Yang et al., 2024). The idea is to rewrite a passage using another LLM, typically ChatGPT, and then assess the similarity or overlap between the original and recomposed text. Passages with larger similarity or overlap will be regarded as LLM-generated. Our approach similarly adheres to the principle of text reconstruction. But it considers only random token deletion and does not require additional API calls, thus is more efficient and economical.

Besides binary classification, some recent studies consider more challenging LLM-generated text detection tasks, e.g., detecting mixtures of human-written and LLM-modified text (Wang et al., 2023a) and tracing the origin of text generation (Li et al., 2023a). These topics are out of the scope of this paper and will be studied as future work.

3 Methodology

This section presents TOCSIN, a novel paradigm to improve zero-shot detection of LLM-generated text. Below we formally define the task in Section 3.1, then illustrate our key assumption in Section 3.2, followed by the detailed approach in Section 3.3.

3.1 Problem Formulation

We study zero-shot LLM-generated text detection, which is formulated as a binary classification problem. Given a piece of text, or candidate passage x, the goal is to discern whether x is human-written or generated by a source LLM. The problem is zero-shot in the sense that we do not assume access to any labeled samples to perform detection.

The method we propose is an enhancement to existing zero-shot detectors. Besides the requirements of the base detector, it also makes use of a semantic similarity measurement model, e.g., BARTScore (Yuan et al., 2021), to calculate token cohesiveness of the candidate passage (see Section 3.2 for details). This model is relatively small and is used off-the-shelf without any fine-tuning.

3.2 Key Assumption

We introduce a new feature, token cohesiveness, to distinguish between LLM-generated and human-written text, and our key assumption is that samples from a source LLM typically exhibit higher token cohesiveness than human-written text. Below we formally define token cohesiveness and state the assumption with an empirical verification.

Definition (Token Cohesiveness). *Given a candidate passage x, let $\tilde{x}$ denote a random copy created by removing a small proportion of tokens from x.*

The token cohesiveness of x is then defined as the expectation of semantic difference between x and $\tilde{x}$, i.e., $u(x) \triangleq \mathbb{E}(\mathrm{DIFF}(x, \tilde{x}))$, where $\mathrm{DIFF}(\cdot, \cdot)$ is a semantic difference metric, and $\mathbb{E}(\cdot)$ the expectation operation.

Token cohesiveness essentially measures the semantic closeness among tokens in the passage. The closer the tokens are semantically related to each other, the higher the token cohesiveness would be. We argue that there is a gap in token cohesiveness between LLM-generated and human-written text. For LLM-generated text, each token is generated based on all its preceding tokens. This would naturally foster a closer relationship among tokens and lead to higher token cohesiveness. But for human-written text, there is no explicit restriction about token generation, potentially resulting in a looser relationship among tokens and, consequently, lower token cohesiveness. We formalize the assertion as a token cohesiveness disparity hypothesis.

Hypothesis (Token Cohesiveness Disparity). *Let $\mathcal{P}_{LLM}$ denote the distribution of LLM-generated text, and $\mathcal{P}_{Human}$ that of human-written text. Then the token cohesiveness $u(x)$ tends to be higher for samples $x \sim \mathcal{P}_{LLM}$, while lower for $x \sim \mathcal{P}_{Human}$.*

We empirically verify the hypothesis in an automated manner. Specifically, as in prior work (Bao et al., 2024; Mitchell et al., 2023), we use 500 news articles randomly sampled from XSum (Narayan et al., 2018) as human-written data, and use the output of four different LLMs when prompted with the first 30 tokens of each human-written article as LLM-generated data. For each article, to calculate its token cohesiveness, we create 10 copies, each randomly deleting 1.5% tokens from the original article. We use negative BARTScore (Yuan et al., 2021) as the semantic difference metric $\mathrm{DIFF}(\cdot, \cdot)$, and approximate the expectation $\mathbb{E}(\cdot)$ with the average of the 10 copies. Figure 1 shows the results, revealing that the token cohesiveness distributions do differ significantly between LLM-generated and human-written data. LLM-generated samples typically show a broader distribution with higher token cohesiveness values.

3.3 Detailed Approach

Based on the above findings, we devise TOCSIN, a generic dual-channel detection paradigm that uses token cohesiveness as a plug-and-play module to improve existing zero-shot detectors. The overall architecture of TOCSIN is sketched in Figure 2.

Algorithm 1 TOCSIN LLM-generated text detection

```
Input: passage $x$, base detector $\mathrm{BASE}(\cdot)$, random token deletion operation $\mathrm{RTD}(\cdot)$, semantic difference metric $\mathrm{DIFF}(\cdot,\cdot)$, decision threshold $\epsilon$
Output: True – LLM-generated, False – human-written
1: $\{\tilde{x}_i\}_{i=1}^{n} \leftarrow \mathrm{RTD}(x)$ for $i = 1, \cdots, n$    ▷ create copies
2: $u(x) \leftarrow \sum_{i=1}^{n} \frac{\mathrm{DIFF}(x, \tilde{x}_i)}{n}$    ▷ token cohesiveness
3: $v(x) \leftarrow \mathrm{BASE}(x)$    ▷ raw prediction
4: $w(x) \leftarrow \begin{cases} e^{u(x)} \times v(x) & \text{if } v(x) \geq 0 \\ e^{-u(x)} \times v(x) & \text{if } v(x) < 0 \end{cases}$    ▷ combination
5: return $w(x) > \epsilon$
```

Specifically, given a passage x, we first feed it into one channel to calculate its token cohesiveness. This channel creates n copies $\{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$ for the input x, with each copy randomly deleting a certain proportion (denoted as ρ) of tokens from x. It then estimates the token cohesiveness of x as:

$$u(x) = \sum_{i=1}^{n} \frac{\mathrm{DIFF}(x, \tilde{x}_i)}{n}, \qquad (1)$$

where $\mathrm{DIFF}(\cdot, \cdot)$ measures the semantic difference between x and each of its copies $\tilde{x}_i$. We employ the established negative BARTScore (Yuan et al., 2021) as the metric, with other evaluated metrics discussed in Appendix A. Meanwhile, we feed the input x into another channel, which is equipped with an existing zero-shot detector, to produce a raw prediction $v(x)$. The output of the two channels are then combined into a new prediction:

$$w(x) = \begin{cases} e^{u(x)} \times v(x) & \text{if} \quad v(x) \geq 0, \\ e^{-u(x)} \times v(x) & \text{if} \quad v(x) < 0, \end{cases} \qquad (2)$$

on which we perform thresholding to make the final decision, i.e., x is categorized as LLM-generated if $w(x) > \epsilon$ or otherwise human-written. This detection procedure is summarized into Algorithm 1.

The proposed dual-channel detection paradigm is rather generic, allowing token cohesiveness to be applied as a plug-and-play module in a variety of detectors to further improve their performance. This paper chooses four state-of-the-art base detectors, including Likelihood (Mitchell et al., 2023), LogRank (Mitchell et al., 2023), LRR (Su et al., 2023) and Fast-DetectGPT (Bao et al., 2024), the details of which are provided in Appendix B.

4 Experiments

This section evaluates the effectiveness of TOCSIN for zero-shot LLM-generated text detection compared with prior state-of-the-arts, and also provides extensive additional analyses to better understand multiple facets of the proposed method.

4.1 Experimental Setups

Datasets To ensure fair comparison, we follow prior work (Bao et al., 2024) to use four diversified datasets: *XSum* for news articles (Narayan et al., 2018), *SQuAD* for Wikipedia content (Rajpurkar et al., 2016), *WritingPrompts* for storytelling (Fan et al., 2018), and *PubMedQA* for biomedical question answering (Jin et al., 2019). Each dataset contains 150 to 500 randomly sampled human-written passages as negative samples, and the same number of LLM-generated passages as positive samples, created by prompting a source model with the first 30 tokens of each negative sample. Eight different source models of various size are considered, including the 1.5B *GPT-2* (Radford et al., 2019), 2.7B *OPT-2.7* (Zhang et al., 2022), 2.7B *GPT-Neo-2.7* (Black et al., 2021), 6B *GPT-J* (Wang and Komatsuzaki, 2021), 20B *GPT-NeoX* (Black et al., 2022), as well as OpenAI's most powerful *ChatGPT* (OpenAI, 2022), *GPT-4* (OpenAI, 2023) and Google's most powerful *Gemini* (Google, 2023) to simulate text generation in real-world scenarios. All these datasets are collected from the open-source project of Fast-DetectGPT (Bao et al., 2024), except for those generated by Gemini, the details of which are introduced in Appendix C.

Metric We also follow prior work to use the area under the receiver operating characteristic curve (*AUROC*) as the evaluation metric. This metric is based on dynamic positive-negative thresholds and does not require specifying a fixed threshold, which is particularly challenging in zero-shot scenarios.

Baselines As we have discussed in Section 3.3, TOCSIN is a generic paradigm that can be applied to various zero-shot detectors to further improve their performance. We choose four zero-shot detectors, *Likelihood* (average log-probability), *LogRank* (average log-rank) (Mitchell et al., 2023), *LRR* (log-probability log-rank ratio) (Su et al., 2023) and *Fast-DetectGPT* (conditional probability curvature) (Bao et al., 2024), and apply TOCSIN on the four detectors to validate its effectiveness and generality. We choose the four detectors as they are recently proposed, computationally efficient, and report current state-of-the-art performance. The derived methods are denoted as $*$+*TOCSIN*.

Besides the four direct baselines, we also compare with other established zero-shot detectors, including *Entropy* (entropy of predictive distribution) (Mitchell et al., 2023), *NPR* (normalized log-rank of perturbations) (Su et al., 2023), *Detect-GPT* (probability curvature) (Mitchell et al., 2023), *DNA-GPT* (divergent n-grams in rewrites) (Yang et al., 2024). We also perform comparisons to supervised classifiers, including the GPT-2 detectors based on *RoBERTa-base/large* (Liu et al., 2019) crafted by OpenAI and *GPTZero* (Tian and Cui, 2023).

Implementation TOCSIN is conceptually simple and easy to implement. For the base detectors, we use the code of Likelihood[1], LogRank[2], LRR[3] and Fast-DetectGPT[4], and keep their settings unchanged. For token cohesiveness calculation, there are two hyperparameters: n (the number of copies created for each input) and ρ (the proportion of tokens to be deleted in each copy). We empirically set $n = 10$, $\rho = 1.5\%$ for **all experiments without re-tuning**. The impact of the two hyperparameters is discussed in Section 4.4.

4.2 Experiments with Open-Source LLMs

We first evaluate TOCSIN in detecting text generated by the five open-source LLMs, from GPT-2 (1.5B) to GPT-NeoX (20B). We use 500 human samples along with 500 LLM samples generated by each of the five models across XSum, SQuAD, and WritingPrompts. By following previous work (Bao et al., 2024), we consider two evaluation settings: (1) the *white-box setting* where each source model is used to score passages, and (2) the *black-box setting* where the source models are not accessible and a surrogate model, i.e., GPT-Neo-2.7, is used to score passages. Note that this scoring is required by the base detectors, not by token cohesiveness calculation. Table 1 presents average AUROC across the three datasets in the two settings, with detailed per dataset results reported in Appendix D.1.

Results in White-Box Setting As we can see, TOCSIN performs particularly well in this setting. It outperforms the four direct baselines (and all the other baselines), irrespective of which dataset or source model is used. The absolute improvement in average AUROC reaches 10.97% over Likelihood, 8.26% over LogRank, and 5.88% over LRR. For Fast-DetectGPT, which already gets a super high average AUROC of 0.9887, TOCSIN still brings consistent and meaningful improvements, pushing the average AUROC further to 0.9946.

[1]https://github.com/eric-mitchell/detect-gpt
[2]https://github.com/baoguangsheng/fast-detect-gpt/
[3]https://github.com/mbzuai-nlp/DetectLLM
[4]https://github.com/baoguangsheng/fast-detect-gpt/

Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
The White-Box Setting						
Entropy (Mitchell et al., 2023)	0.5174	0.4830	0.4898	0.5005	0.5333	0.5048
DNA-GPT (Yang et al., 2024)†	0.9024	0.8797	0.8690	0.8227	0.7826	0.8513
DetectGPT (Mitchell et al., 2023)†	0.9917	0.9758	0.9797	0.9353	0.8943	0.9554
NPR (Su et al., 2023)†	0.9948	0.9832	0.9883	0.9500	0.9065	0.9645
Likelihood (Mitchell et al., 2023)	0.9125	0.8963	0.8900	0.8480	0.7946	0.8683
Likelihood+TOCSIN (ours)	**0.9905**	**0.9876**	**0.9794**	**0.9776**	**0.9549**	**0.9780**
(Absolute ↑)	*7.80%*	*9.13%*	*8.94%*	*12.96%*	*16.03%*	*10.97%*
LogRank (Mitchell et al., 2023)	0.9385	0.9223	0.9226	0.8818	0.8313	0.8993
LogRank+TOCSIN (ours)	**0.9933**	**0.9907**	**0.9857**	**0.9811**	**0.9586**	**0.9819**
(Absolute ↑)	*5.48%*	*6.84%*	*6.31%*	*9.93%*	*12.73%*	*8.26%*
LRR (Su et al., 2023)	0.9601	0.9401	0.9522	0.9179	0.8793	0.9299
LRR+TOCSIN (ours)	**0.9919**	**0.9929**	**0.9873**	**0.9914**	**0.9799**	**0.9887**
(Absolute ↑)	*3.18%*	*5.28%*	*3.51%*	*7.35%*	*10.06%*	*5.88%*
Fast-DetectGPT (Bao et al., 2024)	0.9967	0.9908	0.9940	0.9866	0.9754	0.9887
Fast-DetectGPT+TOCSIN (ours)	**0.9986**	**0.9960**	**0.9978**	**0.9941**	**0.9863**	**0.9946**
(Absolute ↑)	*0.19%*	*0.52%*	*0.38%*	*0.75%*	*1.09%*	*0.59%*
The Black-Box Setting						
DetectGPT (Mitchell et al., 2023)†	0.8517	0.8390	0.9797	0.8575	0.8400	0.8736
Likelihood (ours)	0.7625	0.7838	0.8899	0.8054	0.7851	0.8053
Likelihood+TOCSIN (ours)	**0.9626**	**0.9723**	**0.9794**	**0.9722**	**0.9628**	**0.9699**
(Absolute ↑)	*20.01%*	*18.85%*	*8.95%*	*16.68%*	*17.77%*	*16.46%*
LogRank (ours)	0.8013	0.8210	0.9226	0.8362	0.8070	0.8376
LogRank+TOCSIN (ours)	**0.9644**	**0.9753**	**0.9857**	**0.9737**	**0.9630**	**0.9724**
(Absolute ↑)	*16.31%*	*15.43%*	*6.31%*	*13.75%*	*15.60%*	*13.48%*
LRR (ours)	0.8505	0.8609	0.9518	0.8637	0.8187	0.8691
LRR+TOCSIN (ours)	**0.9745**	**0.9849**	**0.9873**	**0.9832**	**0.9752**	**0.9810**
(Absolute ↑)	*12.40%*	*12.40%*	*3.55%*	*11.95%*	*15.65%*	*11.19%*
Fast-DetectGPT (Bao et al., 2024)	0.9834	0.9572	0.9984	0.9592	0.9404	0.9677
Fast-DetectGPT+TOCSIN (ours)	**0.9948**	**0.9815**	**0.9994**	**0.9822**	**0.9741**	**0.9864**
(Absolute ↑)	*1.14%*	*2.43%*	*0.10%*	*2.30%*	*3.37%*	*1.87%*

Table 1: AUROC for zero-shot detection of passages generated from five source models, averaged across XSum, SQuAD, WritingPrompts, with detailed results provided in Appendix D.1. Results marked by "(ours)" are produced by ourselves, and other results are taken directly from (Bao et al., 2024) to avoid implementation bias. In the white-box (resp. black-box) setting, the source model (resp. GPT-Neo-2.7) is used for scoring. † denotes methods that invoke scoring models multiple times, thereby significantly increasing computational demands. **Bold** indicates that a +TOCSIN variant outperforms its direct baseline, and "*(Absolute ↑)*" the absolute improvements.

Results in Black-Box Setting In this setting we observe similar phenomena. TOCSIN, again, consistently outperforms all the baselines. Notably, the improvements are even more significant than those in the white-box setting (the absolute boosts in average AUROC reach 16.46%, 13.48%, 11.19%, and 1.87% over the four direct baselines). We speculate this is because the base detectors require the source model to make predictions. In the black-box setting where the source model is not available, they have to use a surrogate model for approximation, which inevitably results in performance degradation. TOCSIN, in contrast, requires no such approximation and can therefore resist the degradation.

4.3 Experiments with API-based LLMs

We further evaluate TOCSIN in detecting passages generated by ChatGPT, GPT-4, and Gemini to simulate real-world scenarios. We use 150 positive and 150 negative samples on XSum, WritingPrompts, and PubMedQA. As we are not able to access the source models, we consider the black-box setting, with GPT-Neo-2.7 as the surrogate model. Table 2 reports the results, showing that TOCSIN can bring consistent improvements to the four direct baselines in almost all cases. The only exception is the PubMedQA data produced by ChatGPT or GPT-4. In these two cases we find that PubMedQA passages, which consist of only 64 tokens on average, are substantially shorter than passages on the other datasets (e.g., 221 tokens on XSum and 218 tokens on WritingPrompts). For LLM-generated text, the generation of each token is an aggregation process, making the generated token more closely related to its preceding tokens. Passages of shorter length undergo fewer such aggregation processes, which may suppress the closeness of tokens therein and reduce token cohesiveness, making it more difficult

Method	ChatGPT			GPT-4			Gemini		
	XSum	**Writing**	**PubMed**	**XSum**	**Writing**	**PubMed**	**XSum**	**Writing**	**PubMed**
Supervised Classifiers									
RoBERTa-base (Bao et al., 2024)	0.9150	0.7084	0.6188	0.6778	0.5068	0.5309	0.8708	0.8002	0.4460
RoBERTa-large (Bao et al., 2024)	0.8507	0.5480	0.6731	0.6879	0.3821	0.6067	0.8101	0.6296	0.4508
GPTzero (Tian and Cui, 2023)	0.9952	0.9292	0.8799	0.9815	0.8262	0.8482	0.9987	0.9837	0.8840
Zero-Shot Classifiers									
Entropy (Mitchell et al., 2023)	0.3305	0.1902	0.2767	0.4360	0.3702	0.3295	0.5399	0.4395	0.4335
DNA-GPT (Yang et al., 2024)†	0.9124	0.9425	0.7959	0.7347	0.8032	0.7565	0.8675	0.9257	0.5199
DetectGPT (ours)†	0.8901	0.9452	0.6362	0.6692	0.8177	0.5927	0.7549	0.9151	0.6854
NPR (Su et al., 2023)†	0.7899	0.8924	0.6784	0.5280	0.6122	0.6328	0.8172	0.9487	0.6384
Likelihood (Mitchell et al., 2023)	0.9578	0.9740	**0.8775**	0.7980	0.8553	**0.8104**	0.8519	0.9114	0.7616
Likelihood+TOCSIN (ours)	**0.9984**	**0.9933**	0.8701	**0.9736**	**0.9324**	0.8044	**0.8691**	**0.9256**	**0.9823**
(Absolute ↑)	*4.06%*	*1.93%*	*-0.74%*	*17.56%*	*7.71%*	*-0.60%*	*1.72%*	*1.42%*	*22.07%*
LogRank (Mitchell et al., 2023)	0.9582	0.9656	**0.8687**	0.7975	0.8286	**0.8003**	0.8628	0.9076	0.7689
LogRank+TOCSIN (ours)	**0.9981**	**0.9933**	0.8620	**0.9716**	**0.9208**	0.7952	**0.8655**	**0.9175**	**0.9716**
(Absolute ↑)	*3.99%*	*2.77%*	*-0.67%*	*17.41%*	*9.22%*	*-0.51%*	*0.27%*	*0.99%*	*20.27%*
LRR (Su et al., 2023)	0.9162	0.8958	**0.7433**	0.7447	0.7028	**0.6814**	0.7274	0.8179	0.7234
LRR+TOCSIN (ours)	**0.9939**	**0.9927**	0.7092	**0.9614**	**0.8036**	0.6465	**0.8720**	**0.9195**	**0.9978**
(Absolute ↑)	*7.77%*	*9.69%*	*-3.41%*	*21.67%*	*10.08%*	*-3.49%*	*14.46%*	*10.16%*	*27.44%*
Fast-DetectGPT (Bao et al., 2024)	0.9907	0.9916	**0.9021**	0.9067	0.9612	**0.8503**	0.8518	0.9465	0.8769
Fast-DetectGPT+TOCSIN (ours)	**0.9969**	**0.9964**	0.9011	**0.9455**	**0.9708**	0.8490	**0.8697**	**0.9484**	**0.9799**
(Absolute ↑)	*0.62%*	*0.48%*	*-0.10%*	*3.88%*	*0.96%*	*-0.13%*	*1.79%*	*0.19%*	*10.30%*

Table 2: AUROC for detecting passages generated by ChatGPT, GPT-4, and Gemini. For ChatGPT and GPT-4, results marked by "(ours)" are produced by ourselves, and other results are taken directly from (Bao et al., 2024) to avoid implementation bias. For Gemini, all results are produced by ourselves. The black-box setting is used for all zero-shot classifiers, with GPT-Neo-2.7 as surrogate model. **Bold** stands for better performance between a baseline and its +TOCSIN version, and "*(Absolute ↑)*" the absolute improvements.

to distinguish these passages from human-written ones via token cohesiveness. We will show later in Section 4.4 the detailed impact of passage length.

4.4 Additional Analyses

We provide additional analyses to better understand multiple facets of TOCSIN. We consider only the more practical black-box setting where the source LLMs are not accessible.

Time & Space Efficiency As an improvement to existing zero-shot detectors, TOCSIN keeps the base detector unchanged, and the additional time and space costs mainly come from the computation of token cohesiveness, which involves 10 rounds of random token deletion and BARTScore computation. To rigorously assess the additional costs, we conduct time and space analysis on XSum, SQuAD, and WritingPrompts where the LLM-generated passages are produced by the five open-source models, and we make comparison between Fast-DetectGPT and Fast-DetectGPT+TOCSIN. All experiments here are conducted on a single NVIDIA A40 GPU with 48GB memory. Table 3 reports the average runtime per instance and GPU memory usage of the two variants, with detailed results further provided in Appendix D.2. Compared to the base detector,

	Runtime (s)	**GPU Memory (GB)**
w/o TOCSIN	0.31	23.96
w/ TOCSIN	0.47	28.67
(Absolute ↑)	*0.16*	*4.71*

Table 3: Runtime per instance and GPU memory usage of w/ and w/o TOCSIN variants of Fast-DetectGPT in black-box setting, averaged across five source models on XSum, SQuAD, and WritingPrompts. "*(Absolute ↑)*" means additional time/space cost brought by TOCSIN.

TOCSIN brings a relatively low additional runtime of 0.16s per instance and a relatively low additional GPU memory usage of 4.71GB on average.

Complementarity with Existing Detectors The success of TOCSIN is largely attributed to the good complementarity between the token cohesiveness feature and existing detectors. To validate this viewpoint, we compare the performance of combining Fast-DetectGPT with an existing detector versus combining it with token cohesiveness, in the same manner as described in Section 3.3. Table 4 reports average AUROC across XSum, SQuAD, WritingPrompts for the five open-source models, showing that combining Fast-DetectGPT with Likelihood, LogRank, or LRR does not always yield substantial improvements as it does with TOCSIN.

Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
Fast-DetectGPT	0.9834	0.9572	0.9984	0.9592	0.9404	0.9677
Fast-DetectGPT+Likelihood	0.9696	0.9472	0.9967	0.9522	0.9316	0.9595
Fast-DetectGPT+LogRank	0.9786	0.9558	0.9982	0.9589	0.9389	0.9661
Fast-DetectGPT+LRR	0.9833	0.9598	0.9985	0.9612	0.9420	0.9690
Fast-DetectGPT+TOCSIN	**0.9948**	**0.9815**	**0.9994**	**0.9822**	**0.9741**	**0.9864**

Table 4: AUROC for zero-shot detection of passages generated from five source models in the black-box setting, using GPT-Neo-2.7 as the surrogate model. The results are achieved by combining Fast-DetectGPT with different detectors, and averaged across XSum, SQuAD, and WritingPrompts.

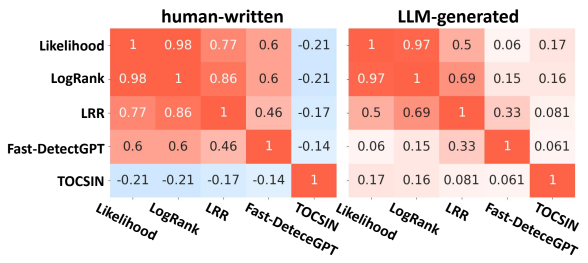

Figure 3: Heatmaps of Pearson Correlation Coefficient between scores from different detectors, averaged across XSum, SQuAD, WritingPrompts and five open-source models. Lighter colors indicate lower correlation, while darker colors indicate stronger correlation.

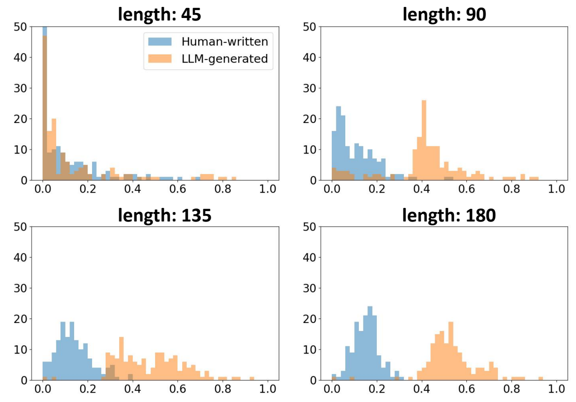

Figure 4: Distribution of token cohesiveness between 150 human-written and 150 ChatGPT-generated passages from WritingPrompts truncated to target length.

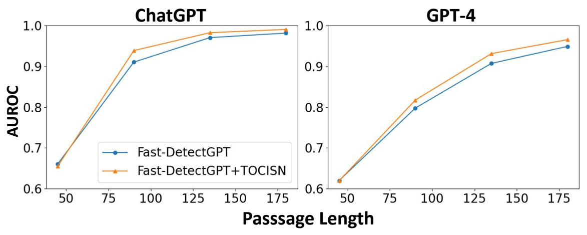

Figure 5: AUROC for detecting ChatGPT and GPT-4 passages on WritingPrompts truncated to target length.

We further examine the Pearson Correlation Coefficient between the Likelihood, LogRank, LRR, Fast-DetectGPT, and token cohesiveness scores for the LLM-generated and human-written passages therein. Figure 3 visualizes the results, averaged across the three datasets and five source models. From the figure, we can observe a relatively high positive correlation among existing detectors, particularly for human-written text, whereas their correlation with the token cohesiveness scores is rather low. This observation indicates strong complementarity between token cohesiveness and existing detectors, which we think is key to the success of our dual-channel detection paradigm. Note that similar to Likelihood, LogRank, LRR and Fast-DetectGPT, token cohesiveness can also be used alone for zero-shot detection of LLM-generated text, the performance of which is shown in Appendix D.3.

Impact of Passage Length We have observed in Table 2 that TOCSIN may not perform that well on shorter passages. To rigorously evaluate the impact of passage length, we truncate the passages from WritingPrompts to various target lengths of 45, 90, 135, 180, and explore how the token cohesiveness and overall performance of TOCSIN varies at the four different lengths. The results are reported in Figure 4 and Figure 5. We can see that at a shorter length of 45, the distributions of token cohesiveness between human-written and LLM-generated text overlap significantly and cannot be discriminated. TOCSIN also fails to improve FastDetect-GPT at this passage length. But as the length increases, the disparities in token cohesiveness between the two types of text become more obvious, and incorporating token cohesiveness into FastDetect-GPT starts to achieve consistent improvements. These results suggest that TOCSIN is more suitable for detecting long passages generated by LLMs.

Impact of Hyperparameters TOCSIN gets two hyperparameters: the number of copies created for each input (n) and the proportion of tokens deleted in each copy (ρ). We examine the impact of the two

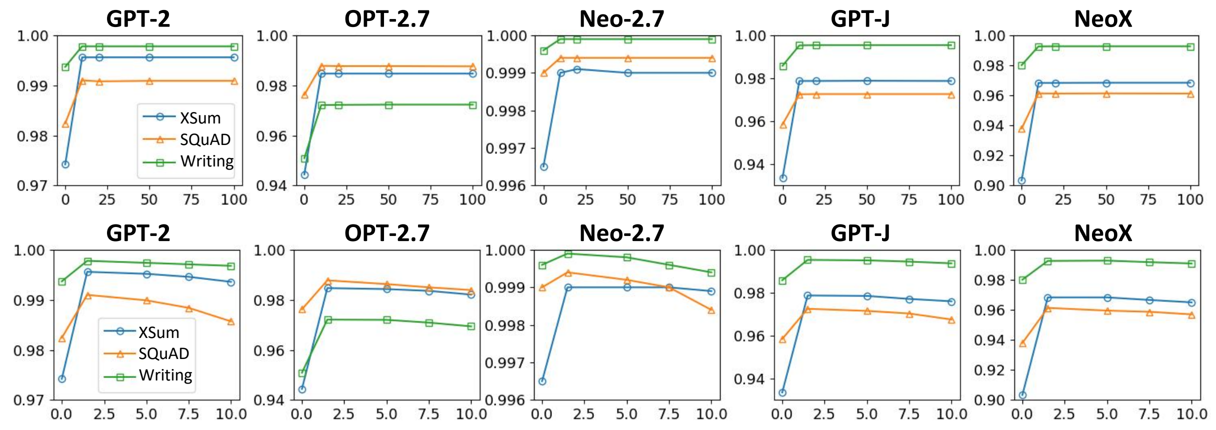

Figure 6: AUROC of FastDetectGPT+TOCSIN with varying hyperparameters on XSum, SQuAD, and Writing-Prompts across five open-source LLMs in the black-box setting. **Top:** number of copies $n \in \{10, 20, 50, 100\}$ and token deletion proportion $\rho = 1.5\%$ where $n = 0$ denotes AUROC of Fast-DetectGPT. **Bottom:** $\rho \in \{1.5\%, 5.0\%, 7.5\%, 10.0\%\}$ and $n = 10$ where $\rho = 0.0\%$ denotes AUROC of Fast-DetectGPT.

hyperparameters, and show how the performance of Fast-DetectGPT+TOCSIN varies as $n \in \{10, 20, 50, 100\}$ and $\rho \in \{1.5\%, 5.0\%, 7.5\%, 10.0\%\}$. Figure 6 presents the results on XSum, SQuAD, and WritingPrompts with LLM-generated passages from the five open-source models, where $n = 0$ and $\rho = 0.0\%$ denote the performance of the base detector Fast-DetectGPT. The results suggest that the performance of TOCSIN is not sensitive to the hyperparameters. TOCSIN performs rather stably with different n values, so we just use $n = 10$ for simplicity and efficiency. Moreover, a smaller ρ value of 1.5% generally performs better, and the performance drops slightly as ρ grows up.

5 Conclusion

This paper introduces the concept of token cohesiveness and unveils that it can serve as a new criterion to discriminate LLM-generated and human-written text. Based on this new finding, we devise TOCSIN for zero-shot detection of LLM-generated text. TOCSIN is a generic dual-channel detection paradigm that uses token cohesiveness as a plug-and-play module to improve existing zero-shot detectors. As empirical evaluation, we apply TOCSIN to four current state-of-the-art base detectors, and achieve meaningful improvements across four diversified datasets with passages generated from eight different source LLMs, demonstrating the effectiveness and generality of our approach.

Limitations

This work has two limitations. First, the proposed method TOCSIN, like most zero-shot detectors, is more suitable for long text and has limited effectiveness on short text. For example, it consistently performs well on passages consisting of 90 tokens or more, but fails on short passages with only 45 tokens, as illustrated in Figure 5. Second, this work is restricted to the most basic form of LLM-generated text detection, i.e., binary classification of LLM-generated and human-written text. Whether TOCSIN still works in more challenging tasks, such as detecting mixtures of LLM-generated and human-written text and tracing the origin of generation, remains an open question for future research.

Acknowledgements

We would like to thank Xingyu Yao for preparing the Gemini datasets. We would also like to thank the action editor and the reviewers for their insightful and valuable suggestions, which significantly improve the quality of this work. This work is supported by the National Natural Science Foundation of China (grants No. 62376033 and 62232006).

References

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? Learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*.

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. In *The Twelfth International Conference on Learning Representations*.

Amrita Bhattacharjee, Tharindu Kumarage, Raha Moraffah, and Huan Liu. 2023. ConDA: Contrastive domain adaptation for AI-generated text detection. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 598–610.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with Mesh-Tensorflow. *https://github.com/EleutherAI/gpt-neo*.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136.

Yutian Chen, Hao Kang, Vivian Zhai, Liangze Li, Rita Singh, and Bhiksha Raj. 2023. GPT-sentinel: Distinguishing human and ChatGPT generated content. *arXiv preprint arXiv:2305.07969*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 889–898.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2024. GPTScore: Evaluate as you desire. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 6556–6576.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Google. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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*.

Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1689–1701.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822.

Bohan Jiang, Zhen Tan, Ayushi Nirmal, and Huan Liu. 2024. Disinformation detection: An evolving challenge in the age of LLMs. In *Proceedings of the 2024 SIAM International Conference on Data Mining*, pages 427–435.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2567–2577.

Thomas Lavergne, Tanguy Urvoy, and François Yvon. 2008. Detecting fake content with relative entropy scoring. In *Proceedings of the 2008 International Conference on Uncovering Plagiarism, Authorship and Social Software Misuse*, volume 377, pages 27–31.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Linyang Li, Pengyu Wang, Ke Ren, Tianxiang Sun, and Xipeng Qiu. 2023a. Origin tracing and detecting of LLMs. *arXiv preprint arXiv:2304.14072*.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2023b. Deepfake text detection in the wild. *arXiv preprint arXiv:2305.13242*.

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. *arXiv preprint arXiv:1907.11692*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the*

40th International Conference on Machine Learning, pages 24950–24962.

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. *arXiv preprint arXiv:2301.13852*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

OpenAI. 2022. ChatGPT: Optimizing language models for dialogue. *https: //openai.com/blog/chatgpt*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ashwinee Panda, Christopher A. Choquette-Choo, Zhengming Zhang, Yaoqing Yang, and Prateek Mittal. 2024. Teach LLMs to phish: Stealing private information from language models. *arXiv preprint arXiv:2403.00871*.

Mike Perkins. 2023. Academic integrity considerations of AI large language models in the post-pandemic era: ChatGPT and beyond. *Journal of University Teaching & Learning Practice*, 20(2):07.

Jiameng Pu, Zain Sarwar, Sifat Muhammad Abdullah, Abdullah Rehman, Yoonjin Kim, Parantapa Bhattacharya, Mobin Javed, and Bimal Viswanath. 2023. Deepfake text detection: Limitations and opportunities. In *2023 IEEE Symposium on Security and Privacy*, pages 1613–1630.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Juan Diego Rodriguez, Todd Hay, David Gros, Zain Shamsi, and Ravi Srinivasan. 2022. Cross-domain detection of GPT-2-generated technical text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1213–1233.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. 2024. The science of detecting LLM-generated text. *Communications of the ACM*, 67(4):50–59.

Edward Tian and Alexander Cui. 2023. GPTZero: Towards detection of AI-generated text using zero-shot and supervised methods. *https://gptzero.me*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. 2020. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8384–8395.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2023. Ghostbuster: Detecting text ghostwritten by large language models. *arXiv preprint arXiv:2305.15047*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model. *https://github.com/kingoflolz/mesh-transformer-jax*.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023a. SeqXGPT: Sentence-level AI-generated text detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1144–1156.

Quan Wang, Licheng Zhang, Zikang Guo, and Zhendong Mao. 2024. IDEATE: Detecting AI-generated text using internal and external factual structures. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 8556–8568.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, et al. 2023b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. *arXiv preprint arXiv:2305.14902*.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. LLMDet: A large language models detection tool. *arXiv preprint arXiv:2305.15004*.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Ruth Petzold, William Yang Wang, and Haifeng Chen. 2024. DNA-GPT: Divergent n-gram analysis for training-free detection of GPT-generated text. In *The Twelfth International Conference on Learning Representations*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. BARTScore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Wanjun Zhong, Duyu Tang, Zenan Xu, Ruize Wang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Neural deepfake detection with factual structure of text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2461–2470.

Biru Zhu, Lifan Yuan, Ganqu Cui, Yangyi Chen, Chong Fu, Bingxiang He, Yangdong Deng, Zhiyuan Liu, Maosong Sun, and Ming Gu. 2023. Beat LLMs at their own game: Zero-shot LLM-generated text detection via querying ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7470–7483.

A Semantic Difference Metrics

We use the negative BARTScore (Yuan et al., 2021) to measure the semantic difference between input x and its copy $\tilde{x}$ with some tokens randomly deleted in our main experiments, i.e.,

$$\mathrm{DIFF}(x, \tilde{x}) = -\mathrm{BARTScore}(x, \tilde{x}).$$

Besides, we also evaluate another semantic difference metric, negative GPTScore (Fu et al., 2024), and discuss the results in Appendix D.4.

$$\mathrm{DIFF}(x, \tilde{x}) = -\mathrm{GPTScore}(x, \tilde{x}).$$

Below we introduce the two metrics in detail.

BARTScore This metric, built on a BART model (Lewis et al., 2020) parameterized by ϕ, is calculated as the log probability of generating the input x (target text) conditioned on its copy $\tilde{x}$ (source text), which can be factorized as:

$$\mathrm{BARTScore}(x, \tilde{x}) = \sum_{j=1}^{k} \log p_\phi(x_j | x_{<j}, \tilde{x}).$$

Here, k is the total number of tokens in x, x_j the j-th token therein, and $x_{<j}$ the sequence preceding x_j. This score essentially measures the semantic coverage between the source and target text, and its negative value therefore measures their semantic difference. In this paper, we use BART-base which has 139M parameters to compute BARTScore, so as to ensure the high efficiency of token cohesiveness calculation. Note that this model is typically much smaller than the scoring model required by the base detector, i.e., the source model itself in the white-box setting and the surrogate model (e.g., the 2.7B GPT-Neo-2.7) in the black-box setting.

GPTScore This metric is similar to BARTScore but has two differences. First, it is built on a GPT model (Radford et al., 2019), parameterized by ϕ, rather than BART. Furthermore, it considers the generation of the target text (input x) conditioned on a more complex source text $T(x, \tilde{x})$, rather than just on $\tilde{x}$ itself. $T(x, \tilde{x})$ is specified as "$\tilde{x}$ [In other words,] x" in the semantic similarity measurement protocol. Summarizing the above two differences, GPTScore is formally defined as:

$$\mathrm{GPTScore}(x, \tilde{x}) = \sum_{j=1}^{k} \log p_\phi(x_j | x_{<j}, T(x, \tilde{x})).$$

We use GPT-2-small with 117M parameters to compute GPTScore, which is also much smaller than the scoring model required by the base detector.

B Base Detectors

In this paper we choose Likelihood (Mitchell et al., 2023), logRank (Mitchell et al., 2023), LRR (Su et al., 2023) and Fast-DetectGPT (Bao et al., 2024), which are computationally efficient and report current state-of-the-art performance as the base detectors. Each base detector requires a scoring model θ to score passages, which is the source LLM in the white-box setting and the surrogate model in the black-box setting (GPT-Neo-2.7 in this paper). Below we introduce the four base detectors in detail.

Likelihood Given a candidate passage x, Likelihood is formally defined as:

$$\mathrm{Likelihood}(x) = \sum_{j=1}^{k} \log p_\theta(x_j | x_{<j}),$$

where $p_\theta(x_j | x_{<j})$ is the probability of token x_j conditioned on its preceding tokens predicted by

the scoring model θ. LLM-generated passages are supposed to have higher Likelihood scores compared to human-written passages.

LogRank Given a candidate passage x, LogRank is formally defined as:

$$\text{LogRank}(x) = -\sum_{j=1}^{k} \log r_\theta(x_j | x_{<j}),$$

where $r_\theta(x_j|x_{<j})$ denotes the rank of the probability of token x_j conditioned on its preceding tokens predicted by the scoring model θ. LLM-generated text tends to have higher LogRank scores compared to human-written text.

LRR LRR makes use of the Log-Likelihood Log-Rank Ratio to discern LLM-generated and human-written text. Given a candidate passage x, LRR is formally defined as:

$$\text{LRR}(x) = -\frac{\sum_{j=1}^{k} \log p_\theta(x_j|x_{<j})}{\sum_{j=1}^{k} \log r_\theta(x_j|x_{<j})},$$

which can be seen as a combination of Likelihood and LogRank. LLM-generated text is supposed to have higher LRR scores than human-written text.

Fast-DetectGPT Fast-DetectGPT makes use of a conditional probability function to detect LLM-generated text. Given a passage x, it first performs conditional independent sampling from $q_\psi(\cdot|x)$ to create a group of samples $\{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$. This sampling samples alternative word choices at each token conditioned on the fixed passage x without depending on other sampled tokens. Then, it evaluates the conditional probabilities $p_\theta(\tilde{x}|x)$ of these samples and combines them to arrive at a decision:

$$\text{Fast-DetectGPT}(x) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{p_\theta(x|x)}{p_\theta(\tilde{x}|x)}.$$

If the score exceeds a specific threshold, the passage is probably LLM-generated. Fast-DetectGPT invokes the sampling model $q_\psi(\cdot|x)$ once to generate all samples and similarly the scoring model $p_\theta(\cdot|x)$ once to evaluate all samples, and therefore is rather efficient. In the white-box setting, $q_\psi(\cdot|x)$ and $p_\theta(\cdot|x)$ are both set to the source LLM. In the black-box setting, $q_\psi(\cdot|x)$ is set to the 6B GPT-J and $p_\theta(\cdot|x)$ the 2.7B GPT-Neo-2.7. All configurations are identical to those in (Bao et al., 2024).

C Datasets Created by Gemini

We follow exactly the same procedure as in (Bao et al., 2024) to generate samples for XSum, WritingPrompts, and PubMedQA by calling the Gemini API. Specifically, we request chat completions with predefined instructions as follows.

Instruction for XSum
```
{‘system’: ‘You are a News writer.’}
{‘user’: ‘Please write an article with about 150
words starting exactly with: <prefix>’}
```
Instruction for WritingPrompts
```
{‘system’: ‘You are a Fiction writer.’}
{‘user’: ‘Please write an article with about 150
words starting exactly with: <prefix>’}
```
Instruction for PubMedQA
```
{‘system’: ‘You are a Technical writer.’}
{‘user’: ‘Please answer the question in about 50
words. <prefix>’}
```

Here `<prefix>` is a prefix consisting of the initial 30 tokens of a human-written passage, e.g., “Maj Richard Scott, 40, is accused of driving at speeds of up to 95mph (153km/h) in bad weather”, and the response is supposed to start with it. We suppose to create 150 samples for each of the three datasets, but unfortunately fail on 19 samples for XSum and 7 for WritingPrompts, resulting in a total of 131 and 143 samples for these two datasets, respectively.

D Additional Experimental Results

D.1 Detailed Results of Open-Source LLMs

Table 5 presents AUROC for zero-shot detection of passages generated by the five open-source LLMs on three datasets of XSum, SQuAD, and Writing-Prompts in the white-box setting, and Table 6 reports the same results in the black-box setting. As we can see, TOCSIN brings consistent improvements in both settings, regardless of the datasets, source models, or base detectors.

D.2 Details of Time & Space Efficiency

Table 7 reports detailed time/space analysis results of Fast-DetectGPT and Fast-DetectGPT+TOCSIN in black-box setting on XSum, SQuAD, and WritingPrompts, with passages generated by the five open-source LLMs. As TOCSIN always performs 10 rounds of random token deletion and uses BART-base to calculate token cohesiveness without any other requirements, the additional time/space costs are rather stable across datasets and source models (as long as the passages are roughly of equal size). On average, TOCSIN bring an additional runtime

Dataset	Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
XSum	Entropy (Mitchell et al., 2023)	0.5835	0.5071	0.5712	0.5705	0.6035	0.5671
	DNA-GPT (Yang et al., 2024)†	0.8548	0.8168	0.8197	0.7586	0.7167	0.7933
	DetectGPT (Mitchell et al., 2023)†	0.9875	0.9621	0.9914	0.9632	0.9398	0.9688
	NPR (Su et al., 2023)†	0.9891	0.9681	0.9929	0.9566	0.9311	0.9676
	Likelihood (Mitchell et al., 2023)	0.8638	0.8600	0.8609	0.8101	0.7604	0.8310
	Likelihood+TOCSIN (ours)	**0.9943**	**0.9917**	**0.9895**	**0.9864**	**0.9829**	**0.9890**
	(Absolute ↑)	*13.05%*	*13.17%*	*12.86%*	*17.63%*	*22.25%*	*15.80%*
	LogRank (Mitchell et al., 2023)	0.8918	0.8839	0.8949	0.8407	0.7939	0.8610
	LogRank+TOCSIN (ours)	**0.9950**	**0.9932**	**0.9916**	**0.9877**	**0.9838**	**0.9903**
	(Absolute ↑)	*10.32%*	*10.93%*	*9.67%*	*14.70%*	*18.99%*	*12.93%*
	LRR (Su et al., 2023)	0.9179	0.8867	0.9190	0.8592	0.8205	0.8807
	LRR+TOCSIN (ours)	**0.9957**	**0.9968**	**0.9935**	**0.9950**	**0.9926**	**0.9947**
	(Absolute ↑)	*7.78%*	*11.01%*	*7.45%*	*13.58%*	*17.21%*	*11.40%*
	Fast-DetectGPT (Bao et al., 2024)	0.9930	0.9803	0.9885	0.9771	0.9703	0.9818
	Fast-DetectGPT+TOCSIN (ours)	**0.9974**	**0.9928**	**0.9966**	**0.9927**	**0.9850**	**0.9929**
	(Absolute ↑)	*0.44%*	*1.25%*	*0.81%*	*1.56%*	*1.47%*	*1.11%*
SQuAD	Entropy (Mitchell et al., 2023)	0.5791	0.5119	0.5581	0.5643	0.6056	0.5638
	DNA-GPT (Yang et al., 2024)†	0.9094	0.8934	0.8589	0.8069	0.7525	0.8442
	DetectGPT (Mitchell et al., 2023)†	0.9914	0.9763	0.9625	0.8738	0.7916	0.9191
	NPR (Su et al., 2023)†	0.9965	0.9853	0.9789	0.9108	0.8175	0.9378
	Likelihood (Mitchell et al., 2023)	0.9077	0.8839	0.8585	0.7943	0.6977	0.8284
	Likelihood+TOCSIN (ours)	**0.9888**	**0.9733**	**0.9608**	**0.9562**	**0.8935**	**0.9545**
	(Absolute ↑)	*8.11%*	*8.94%*	*10.23%*	*16.19%*	*19.58%*	*12.61%*
	LogRank (Mitchell et al., 2023)	0.9454	0.9203	0.9054	0.8471	0.7545	0.8745
	LogRank+TOCSIN (ours)	**0.9928**	**0.9814**	**0.9739**	**0.9630**	**0.9019**	**0.9626**
	(Absolute ↑)	*4.74%*	*6.11%*	*6.85%*	*11.59%*	*14.74%*	*8.81%*
	LRR (Su et al., 2023)	0.9773	0.9597	0.9610	0.9244	0.8600	0.9365
	LRR+TOCSIN (ours)	**0.9928**	**0.9836**	**0.9777**	**0.9862**	**0.9621**	**0.9805**
	(Absolute ↑)	*1.55%*	*2.39%*	*1.67%*	*6.18%*	*10.21%*	*4.40%*
	Fast-DetectGPT (Bao et al., 2024)	0.9990	0.9949	0.9956	0.9853	0.9617	0.9873
	Fast-DetectGPT+TOCSIN (ours)	**0.9996**	**0.9972**	**0.9975**	**0.9904**	**0.9764**	**0.9922**
	(Absolute ↑)	*0.06%*	*0.23%*	*0.19%*	*0.51%*	*1.47%*	*0.49%*
WritingPrompts	Entropy (Mitchell et al., 2023)	0.3895	0.4299	0.3400	0.3668	0.3908	0.3834
	DNA-GPT (Yang et al., 2024)†	0.9431	0.9288	0.9283	0.9026	0.8786	0.9163
	DetectGPT (Mitchell et al., 2023)†	0.9962	0.9891	0.9852	0.9688	0.9516	0.9782
	NPR (Su et al., 2023)†	0.9987	0.9962	0.9930	0.9825	0.9708	0.9882
	Likelihood (Mitchell et al., 2023)	0.9661	0.9451	0.9505	0.9396	0.9256	0.9454
	Likelihood+TOCSIN (ours)	**0.9884**	**0.9976**	**0.9880**	**0.9901**	**0.9882**	**0.9905**
	(Absolute ↑)	*2.23%*	*5.25%*	*3.75%*	*5.05%*	*6.26%*	*4.51%*
	LogRank (Mitchell et al., 2023)	0.9782	0.9628	0.9675	0.9577	0.9454	0.9623
	LogRank+TOCSIN (ours)	**0.9922**	**0.9976**	**0.9916**	**0.9927**	**0.9902**	**0.9929**
	(Absolute ↑)	*1.40%*	*3.48%*	*2.41%*	*3.50%*	*4.48%*	*3.06%*
	LRR (Su et al., 2023)	0.9850	0.9740	0.9766	0.9702	0.9573	0.9726
	LRR+TOCSIN (ours)	**0.9871**	**0.9983**	**0.9907**	**0.9929**	**0.9850**	**0.9908**
	(Absolute ↑)	*0.21%*	*2.43%*	*1.41%*	*2.27%*	*2.77%*	*1.82%*
	Fast-DetectGPT (Bao et al., 2024)	0.9982	0.9972	0.9980	0.9974	0.9941	0.9970
	Fast-DetectGPT+TOCSIN (ours)	**0.9988**	**0.9979**	**0.9993**	**0.9992**	**0.9974**	**0.9985**
	(Absolute ↑)	*0.06%*	*0.07%*	*0.13%*	*0.18%*	*0.33%*	*0.15%*

Table 5: Details of the main results in Table 1 on three datasets in **white-box setting**, with all setups identical to those in Table 1.

of 0.16s per instance and additional GPU memory usage of 4.71GB.

D.3 TOCSIN as A Standalone Metric

We evaluate TOCSIN as a standalone metric and compare it with our baselines. Since TOCSIN is a fully black-box detector (which even does not require a surrogate model), we compare their performance under the black-box setting. Table 8 shows the average AUROC across XSum, SQuAD, and WritingPrompts. The results reveal that TOCSIN, when used alone, outperforms all competitive baselines except the current best Fast-DetectGPT.

We further examine the score distributions to understand why it trails behind Fast-DetectGPT. We find that unlike Fast-DetectGPT, where human-written and LLM-generated text scores are almost separate with minimal overlap (Bao et al., 2024, Figure 1), TOCSIN scores for human-written text fall largely within the range of LLM-generated text

Dataset	Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
XSum	DetectGPT (Mitchell et al., 2023)†	0.9180	0.8868	0.9914	0.8830	0.8682	0.9095
	Likelihood (ours)	0.7308	0.7918	0.8609	0.7508	0.7429	0.7754
	Likelihood+TOCSIN (ours)	**0.9901**	**0.9891**	**0.9895**	**0.9829**	**0.9825**	**0.9868**
	(Absolute ↑)	*25.93%*	*19.73%*	*12.86%*	*23.21%*	*23.96%*	*21.14%*
	LogRank (ours)	0.7610	0.8139	0.8950	0.7747	0.7550	0.7999
	LogRank+TOCSIN (ours)	**0.9887**	**0.9901**	**0.9916**	**0.9824**	**0.9807**	**0.9867**
	(Absolute ↑)	*22.77%*	*17.62%*	*9.66%*	*20.77%*	*22.57%*	*18.68%*
	LRR (ours)	0.7824	0.8069	0.9186	0.7767	0.7357	0.8041
	LRR+TOCSIN (ours)	**0.9904**	**0.9953**	**0.9935**	**0.9914**	**0.9901**	**0.9921**
	(Absolute ↑)	*20.80%*	*18.84%*	*7.49%*	*21.47%*	*25.44%*	*18.80%*
	Fast-DetectGPT (Bao et al., 2024)	0.9742	0.9444	0.9965	0.9335	0.9033	0.9504
	Fast-DetectGPT+TOCSIN (ours)	**0.9956**	**0.9847**	**0.9990**	**0.9787**	**0.9683**	**0.9853**
	(Absolute ↑)	*2.14%*	*4.03%*	*0.25%*	*4.52%*	*6.50%*	*3.49%*
SQuAD	DetectGPT (Mitchell et al., 2023)†	0.7382	0.7530	0.9625	0.7882	0.7709	0.8026
	Likelihood (ours)	0.6772	0.7372	0.8584	0.7562	0.7206	0.7499
	Likelihood+TOCSIN (ours)	**0.9382**	**0.9346**	**0.9608**	**0.9480**	**0.9229**	**0.9409**
	(Absolute ↑)	*26.10%*	*19.74%*	*10.24%*	*19.18%*	*20.23%*	*19.10%*
	LogRank (ours)	0.7387	0.7877	0.9052	0.8042	0.7579	0.7987
	LogRank+TOCSIN (ours)	**0.9378**	**0.9437**	**0.9739**	**0.9512**	**0.9243**	**0.9462**
	(Absolute ↑)	*19.91%*	*15.60%*	*6.87%*	*14.7%*	*16.64%*	*14.75%*
	LRR (ours)	0.8447	0.8615	0.9603	0.8709	0.8109	0.8697
	LRR+TOCSIN (ours)	**0.9713**	**0.9620**	**0.9777**	**0.9701**	**0.9552**	**0.9673**
	(Absolute ↑)	*12.66%*	*10.05%*	*1.74%*	*9.92%*	*14.43%*	*9.76%*
	Fast-DetectGPT (Bao et al., 2024)	0.9824	0.9762	0.9990	0.9584	0.9379	0.9708
	Fast-DetectGPT+TOCSIN (ours)	**0.9910**	**0.9878**	**0.9994**	**0.9725**	**0.9613**	**0.9824**
	(Absolute ↑)	*0.86%*	*1.16%*	*0.04%*	*1.41%*	*2.34%*	*1.16%*
WritingPrompts	DetectGPT (Mitchell et al., 2023)†	0.8989	0.8772	0.9852	0.9014	0.8809	0.9087
	Likelihood (ours)	0.8795	0.8225	0.9505	0.9093	0.8919	0.8907
	Likelihood+TOCSIN (ours)	**0.9596**	**0.9933**	**0.9880**	**0.9857**	**0.9831**	**0.9819**
	(Absolute ↑)	*8.01%*	*17.08%*	*3.75%*	*7.64%*	*9.12%*	*9.12%*
	LogRank (ours)	0.9043	0.8614	0.9675	0.9298	0.9081	0.9142
	LogRank+TOCSIN (ours)	**0.9667**	**0.9922**	**0.9916**	**0.9874**	**0.9839**	**0.9844**
	(Absolute ↑)	*6.24%*	*13.08%*	*2.41%*	*5.76%*	*7.58%*	*7.02%*
	LRR (ours)	0.9244	0.9142	0.9766	0.9436	0.9095	0.9337
	LRR+TOCSIN (ours)	**0.9617**	**0.9975**	**0.9907**	**0.9880**	**0.9803**	**0.9836**
	(Absolute ↑)	*3.73%*	*8.33%*	*1.41%*	*4.44%*	*7.08%*	*4.99%*
	Fast-DetectGPT (Bao et al., 2024)	0.9937	0.9509	0.9996	0.9858	0.9801	0.9820
	Fast-DetectGPT+TOCSIN (ours)	**0.9978**	**0.9721**	**0.9999**	**0.9953**	**0.9926**	**0.9915**
	(Absolute ↑)	*0.41%*	*2.12%*	*0.03%*	*0.95%*	*1.25%*	*0.95%*

Table 6: Details of the main results in Table 1 on three datasets in **black-box setting**, with all setups identical to those in Table 1.

Dataset	Method	Runtime (s)					GPU Memory (GB)				
		GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX
XSum	w/o TOCSIN	0.31	0.31	0.31	0.31	0.31	23.57	23.30	23.22	24.21	23.53
	w/ TOCSIN	0.48	0.46	0.46	0.47	0.46	28.35	27.80	28.37	28.79	27.35
	(Absolute ↑)	*0.17*	*0.15*	*0.15*	*0.16*	*0.15*	*4.78*	*4.50*	*5.15*	*4.58*	*3.82*
SQuAD	w/o TOCSIN	0.32	0.32	0.32	0.32	0.31	23.35	23.98	23.90	24.05	24.00
	w/ TOCSIN	0.49	0.46	0.49	0.48	0.47	28.14	28.53	28.54	28.07	29.19
	(Absolute ↑)	*0.17*	*0.14*	*0.17*	*0.16*	*0.16*	*4.79*	*4.55*	*4.64*	*4.02*	*5.19*
Writing	w/o TOCSIN	0.32	0.30	0.31	0.31	0.31	24.98	24.30	23.89	24.75	24.35
	w/ TOCSIN	0.48	0.45	0.48	0.47	0.47	29.31	29.24	29.51	29.56	29.32
	(Absolute ↑)	*0.16*	*0.15*	*0.17*	*0.16*	*0.16*	*4.33*	*4.94*	*5.62*	*4.81*	*4.97*

Table 7: Runtime per instance and GPU memory usage of w/ and w/o TOCSIN variants of Fast-DetectGPT in black-box setting. "*(Absolute ↑)*" means additional time/space cost brought by TOCSIN.

scores , as shown in Figure 1. This makes it particularly challenging to identify LLM-generated text with very low TOCSIN scores, as these scores fall perfectly within the range for human-written text.

Moreover, we would like to emphasize that TOCSIN's lesser performance when used alone, com-

Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
Likelihood	0.7625	0.7838	0.8899	0.8054	0.7851	0.8053
LogRank	0.8013	0.8210	0.9226	0.8362	0.8070	0.8376
LRR	0.8505	0.8609	0.9518	0.8637	0.8187	0.8691
DetectGPT	0.8517	0.8390	0.9797	0.8575	0.8400	0.8736
Fast-DetectGPT	0.9834	0.9572	0.9984	0.9592	0.9404	0.9677
TOCSIN	0.9307	0.9518	0.9188	0.9424	0.9357	0.9359

Table 8: AUROC of Likelihood, LogRank, LRR, DetectGPT, Fast-DetectGPT, and TOCSIN used as a standalone metric. The black-box setting is used for all zero-shot classifiers, with GPT-Neo-2.7 as surrogate model. The results are averaged across XSum, SQuAD, and WritingPrompts, with other settings identical to those in Table 1.

Method	GPT-2	OPT-2.7	Neo-2.7	GPT-J	NeoX	Avg.
The White-Box Setting						
LRR	0.9601	0.9401	0.9522	0.9179	0.8793	0.9299
LRR+TOCSIN (GPTScore)	**0.9631**	**0.9517**	**0.9749**	**0.9342**	**0.9194**	**0.9487**
(Absolute ↑)	*0.30%*	*1.16%*	*2.27%*	*1.63%*	*4.01%*	*1.88%*
Fast-DetectGPT	0.9967	0.9908	0.9940	0.9866	0.9754	0.9887
Fast-DetectGPT+TOCSIN (GPTScore)	**0.9972**	**0.9918**	**0.9951**	**0.9880**	**0.9772**	**0.9899**
(Absolute ↑)	*0.05%*	*0.10%*	*0.11%*	*0.14%*	*0.18%*	*0.12%*
The Black-Box Setting						
LRR	0.8505	0.8609	0.9518	0.8637	0.8187	0.8691
LRR+TOCSIN (GPTScore)	**0.9242**	**0.9399**	**0.9749**	**0.9319**	**0.9097**	**0.9361**
(Absolute ↑)	*7.37%*	*7.90%*	*2.31%*	*6.82%*	*9.10%*	*6.70%*
Fast-DetectGPT	0.9834	0.9572	0.9984	0.9592	0.9404	0.9677
Fast-DetectGPT+TOCSIN (GPTScore)	**0.9859**	**0.9621**	**0.9988**	**0.9639**	**0.9476**	**0.9717**
(Absolute ↑)	*0.25%*	*0.49%*	*0.04%*	*0.47%*	*0.72%*	*0.40%*

Table 9: AUROC of LRR, Fast-DetectGPT, and their +TOCSIN versions with token cohesiveness scores computed using GPTScore. During token cohesiveness calculation, the number of copies is fixed at $n = 10$ and the token deletion proportion decreases to $\rho = 0.1\%$. The results are averaged across XSum, SQuAD, and WritingPrompts, with other settings identical to those in Table 1.

pared to the current best Fast-DetectGPT, does not diminish its value as a plug-and-play module to enhance zero-shot detectors. As we have shown in Section 4.4, TOCSIN's unique strength lies in its ability to complement existing detectors, thereby providing improvements when combined.

D.4 Impact of Semantic Difference Metric

TOCSIN requires a semantic difference metric for token cohesiveness calculation. Besides the negative BARTScore used in the main experiments, we further evaluate the negative GPTScore as another metric. We compute token cohesiveness scores using the new metric, and compare the distributions of the scores between LLM-generated and human-written passages. Figure 7 visualizes the results for the same passages that were used in Figure 1, showing that token cohesiveness scores computed using the new metric still exhibit clear distributional differences between the two types of text.

We further integrate these new token cohesiveness scores into LRR and Fast-DetectGPT, and evaluate their performance on XSum, SQuAD, and WritingPrompts. The results are given in Table 9, showing that, with the new metric GPTScore, TOCSIN can still bring consistent improvements to the base detectors. The absolute improvements in average AUROC reach 1.88%/0.12% in the white-box setting and 6.70%/0.40% in the black-box setting over LRR/Fast-DetectGPT, respectively.

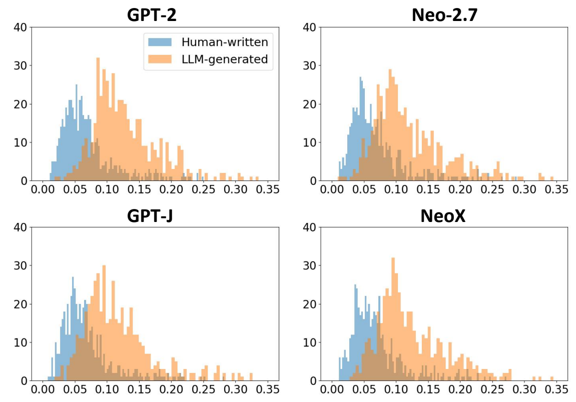

Figure 7: Distributions of token cohesiveness scores computed with GPTScore between human-written and LLM-generated articles. All the articles are identical to those used in Figure 1.