# LAMCL: A Length-aware Momentum Contrastive Learning Framework for Multiscale Machine-Revised Text Detection

**Bing Zhou[1†], Zhe Huang[1†], Shilei Tan[2], Kai Zhao[1], Yongcheng Zhou[1*]**
[1]Chongqing University, Chongqing, China
[2]University of Science and Technology of China, Hefei, China
zhouyongcheng@stu.cqu.edu.cn

## Abstract

Detecting machine-revised text that exhibits subtle lexical differences from the original human-generated text remains a challenge. Recent detection methods, including watermarking-based, logit-based, and training-based models, struggle to capture the fine-grained semantic differences, especially for short texts. To address this issue, we propose **L**ength-**a**ware **M**omentum **C**ontrastive **L**earning (LAMCL), a novel framework for multiscale machine-revised text detection that integrates two core modules. To enhance the discriminative semantic features, the Enhance Before Detection (EBD) module first fuses the original detected text with the counterpart processed by a Large Language Model (LLM), and then measures semantic consistency to distinguish between machine-revised and human-generated text. Meanwhile, based on the Momentum Contrastive Learning (MCL) framework, the Length-aware Weighting (LW) module leverages text length and label information for hard negative sampling, mitigating the ambiguity of short text attribution and boosting the robustness of representation learning. Experimental results demonstrate that our method outperforms the existing detectors in identifying multiscale machine-revised text across diverse practical scenarios, tasks, and LLMs. The code is available at https://github.com/hangtze/LAMCL.

## 1 Introduction

With the emergence of LLMs, natural language generation technology has advanced rapidly. Equipped with massive parameter scales, LLMs exhibit strong representational capabilities, which enable them to generate high-quality, semantically rich, and contextually coherent texts. Thus, the application of LLMs has expanded across various industries, bringing considerable convenience to society.

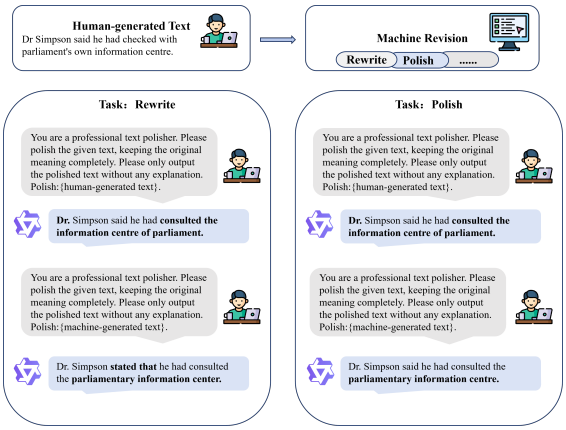



Figure 1: The comparison between the original text and its revision generated by an LLM (Qwen3). The lexical differences between the original human-generated text and the machine-generated text (7 words) are more pronounced than those between the machine-generated text and the machine-regenerated text (5 words or 3 words). The lexical differences are emphasized in **bold.**

However, it also arouses social concerns related to the misuse of LLMs, such as fake news, academic fraud, and exam cheating(Zhang et al., 2024). As a result, it is crucial to distinguish between human-generated texts and machine-generated texts.

Recent detection methods for machine-generated texts mainly include watermarking-based models (Yang et al., 2022; Xiang et al., 2018), logit-based models (Ippolito et al., 2020; Gehrmann et al., 2019; Solaiman et al., 2019; Mitchell et al., 2023; Bao et al., 2024), and training-based models (Chen et al., 2025; Nguyen-Son et al., 2024). While these methods demonstrate considerable efficacy in identifying purely machine-generated texts, their performance in detecting machine-revised texts remains inadequate, particularly for short texts (Zhang et al., 2024). As shown in Figure 1, we randomly select one human-generated text sample from the XSum (Narayan et al., 2018) dataset, and use an LLM with a predefined prompt (see Appendix B) to revise

† Equal contribution. ∗ Corresponding author.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24366–24380

this sample. The distinctions between machine-revised text and human-generated text lie in subtle semantic features(Chen et al., 2025). Consequently, the mainstream detectors fail to accurately identify machine-revised texts (see Section 5.1). In this paper, we aim to devise a novel method that can reliably detect multiscale machine-revised texts.

The machine-generated text and machine-regenerated text share substantial lexical overlap, whereas both exhibit lower similarity with the original human-generated text (Nguyen-Son et al., 2024), as shown in Figure 1. We have the intuition that these differences between the original text and its revised counterpart can help to distinguish between human-generated and machine-revised text. To leverage these differences and enhance discriminative semantic features, our proposed framework LAMCL utilizes the EBD module to process the detected text and concatenate the original text with its LLM-processed counterpart. Subsequently, both the encoded enhanced text and the encoded original text, derived from the main encoder, are fed into the Consistency Measurement (CM) module to measure their semantic consistency, based on which the detected text is identified as either human-generated or machine-revised text. Meanwhile, we fine-tune the framework via the MCL with LW module to further reinforce the framework's ability to capture fine-grained semantic differences. Specifically, the MCL with LW module constructs positive samples via dropout noise, and performs hard negative sampling by leveraging text length and label information for representation learning.

The contributions of this paper can be summarized as follows:

- We propose a novel framework LAMCL to distinguish between human-generated texts and machine-revised texts, which utilizes the EBD module to enhance discriminative semantic features and the MCL framework with the LW module to leverage text length and label information for representation learning.
- We construct a comprehensive benchmark dataset via LLM prompt-based generation to evaluate the robustness of multiscale machine-revised text detection, which covers diverse practical scenarios, tasks, and LLMs.
- We conduct extensive experiments on the benchmark dataset. Experimental results demonstrate that our method outperforms existing detectors, achieving significant improvements in AUROC.

## 2 Related work

### 2.1 Machine-generated Text Detection Methods

The mainstream methods for machine-generated text detection can be classified into three types of models.

Watermarking-based models achieve detection by embedding watermarks into machine-generated texts. Xiang et al. (2018) analyzed relative synonymous word frequencies, and then performed synonym substitution to embed watermarks. They overlooked the fact that such synonym substitution may undermine the contextual semantics. Yang et al. (2022) employed BERT to generate suitable synonym candidates and evaluated the semantic consistency of the text before and after watermark embedding, ensuring that the synonym substitution maintained the text's readability without compromising its core meaning. However, these methods all require modifying the generative model and altering the detected texts, making them impractical for detection in our scenario.

Logit-based models achieve detection based on token probabilities from the generative models. Gehrmann et al. (2019) introduced a visual tool GLTR, offering auxiliary means for manual detection through token probability ranking and entropy analysis. Ippolito et al. (2020) further improved the GLTR detection model by training logistic regression binary classifiers for machine-generated text detection, with input histograms expanded from 4 dimensions to 50 dimensions. Solaiman et al. (2019) introduced a simple baseline method for machine-generated text detection, which relied on a threshold for the total mean log probability of the detected text. Mitchell et al. (2023) introduced a zero-shot DetectGPT, which eliminated the need for explicit training by utilizing random perturbations and computing log probabilities. Fast-DetectGPT (Bao et al., 2024) is an advanced variant of DetectGPT, proposing the concept of conditional probability curvature. It replaced the perturbation step with a more efficient sampling procedure, retaining the zero-shot advantage while improving computational efficiency. However, these methods exhibit limited effectiveness in detecting machine-revised texts that are similar to the original human-

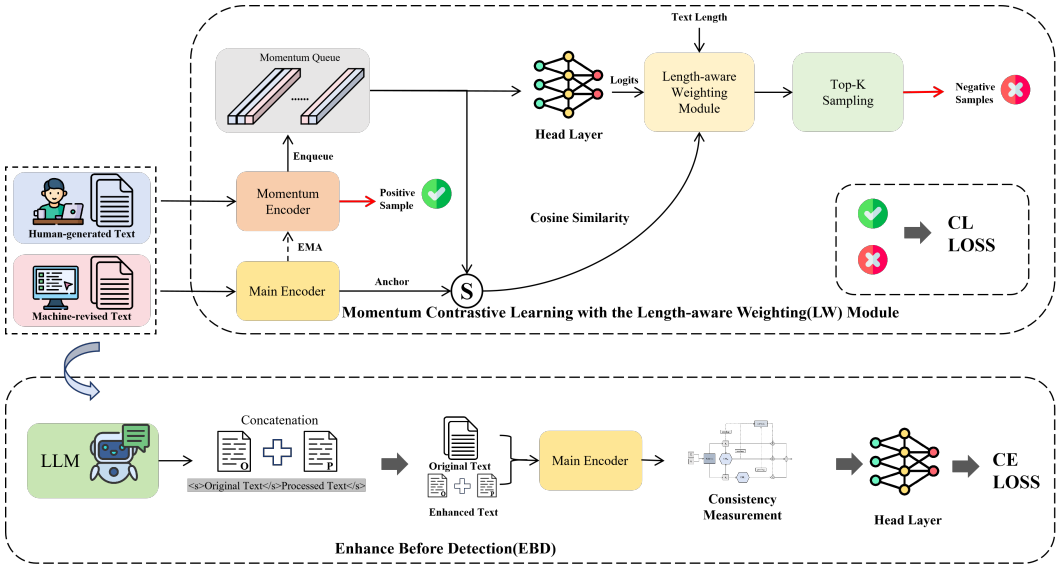



Figure 2: An overview of our proposed method LAMCL. (Below) The EBD module first concatenates the original text with its LLM-processed counterpart. Subsequently, both the encoded enhanced text and the encoded original text, derived from the main encoder, are fed into the CM module to measure their semantic consistency. (Above) In the MCL with the LW module, the positive sample is constructed by the momentum encoder via dropout noise, and the LW module leverages text length and label information for hard negative sampling. The parameters of the momentum encoder are updated solely via the exponential moving average (EMA) mechanism. Contrastive learning (CL) loss and cross-entropy (CE) loss jointly optimize the method for multiscale machine-revised text detection.

generated texts. Thus, Chen et al. (2025) proposed the existing state-of-the-art framework Imitate Before Detect (ImBD) which first imitated the stylistic patterns of machine-revised texts via Style Preference Optimization and then leveraged the Style-Conditional Probability Curvature metric for distributional difference measurement to distinguish between machine-revised and human-generated text.

Training-based models (Chen et al., 2025; Nguyen-Son et al., 2024) depend on labeled data for model fine-tuning. RoBERTa-base and RoBERTa-large (Liu et al., 2019) are optimized variants of the BERT model with enhanced pre-training strategies, widely adopted as backbone architectures for downstream text detection tasks. For machine-revised text detection, SIMLLM (Nguyen-Son et al., 2024) fine-tuned the RoBERTa pre-trained model and distinguished machine-revised texts by leveraging the similarity between input texts and their proofread versions. However, SIMLLM only captures coarse-grained similarity and cannot fully model the fine-grained lexical differences between machine-revised and original human texts. To address these limitations and better learn discriminative features from subtle edits, we propose the EBD module equipped with the CM module, which enables in-depth representation learning of such lexical differences. Specifically, the CM module utilizes multi-head attention to model semantic interactions between the original text and its enhanced counterpart, measuring both cross-consistency and self-consistency. By fusing these consistency-aware features, the model obtains more distinctive representations to accurately identify machine-revised texts.

## 2.2 Contrastive Learning Methods

Contrastive learning (CL) is normally employed for representation learning in such a way that positive samples are pulled together and negative samples are pushed apart (Hadsell et al., 2006; Van Gansbeke et al., 2020; Bachman et al., 2019). In the Computer Vision (CV) field, Chen et al. (2020) adopted random image augmentation to generate negatives and positives. He et al. (2020) proposed Momentum Contrast for unsupervised visual representation learning based on dynamic dictionary look-up, which diminished the gap between unsupervised and supervised representation learning. In the Natural Language Processing (NLP) field, existing methods (Kim et al., 2022; Lu et al., 2023; Chen et al., 2024) typically constructed positives via text augmentation and treat all samples in a batch with different labels as negatives, without further selection. Kim et al. (2024) exploited label information to mine negatives that are close to the

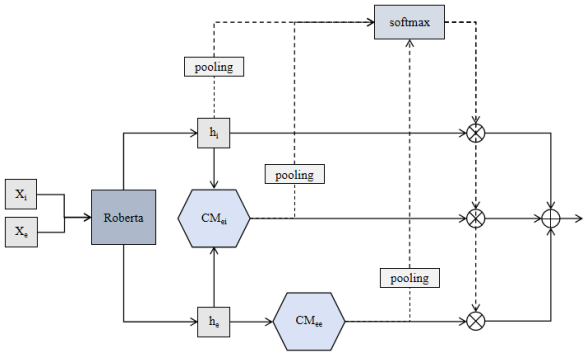



Figure 3: Consistency Measurement

anchor in the MCL framework, but they do not consider that when short texts are used as negatives for representation learning, their attribution ambiguity may degrade CL performance. In contrast, we propose an enhanced MCL framework with the LW module, which constructs positive samples via dropout noise instead of text augmentation, and performs hard negative sampling by leveraging text length and label information. This design jointly addresses semantic distortion from augmentation and ambiguity in the attribution of short texts.

## 3 Methodology

We elaborate on our proposed method LAMCL, focusing on the two core modules: the EBD module and MCL with the LW module. The overview of LAMCL is illustrated in Figure 2, with the main steps detailed as follows:

### 3.1 Enhance Before Detection (EBD)

We propose the EBD module to enhance the semantic features of the detected text before detection. We first adopt an LLM to process the detected text, using the predefined prompts (see Appendix B). We denote the original text as $x_i$ and the processed text as $x_p$. To enhance the semantic features before detection, we concatenate $x_i$ and $x_p$:

$$x_e = [x_i | x_p] \tag{1}$$

where $|$ denotes the concatenation operator and $x_e$ denotes the enhanced text. We concatenate $x_i$ and $x_p$ with the separator token $< /s >$.

We use RoBERTa-base(Liu et al., 2019) as the main encoder to encode $x_e$ and $x_i$, denoted as $h_e$ and $h_i$, respectively. To model the latent interactions between these representations, we introduce the CM module, as shown in Figure 3. The consistency measurement is defined as follows:

$$CM_{ei} = Concat_{h=1}^{H}\left(\sigma\left(\frac{h_e \cdot h_i^{\top}}{\sqrt{d_k}}\right) h_i\right) \tag{2}$$

$$CM_{ee} = Concat_{h=1}^{H}\left(\sigma\left(\frac{h_e \cdot h_e^{\top}}{\sqrt{d_k}}\right) h_e\right) \tag{3}$$

where H denotes the number of heads of the multi-head attention(MHA), $\sigma$ denotes softmax function, and $d_k$ denotes the dimension of the key vectors in the MHA.

To construct the final enhanced representations, we first extract the [CLS] token embeddings from $h_i$, $CM_{ei}$, and $CM_{ee}$, then compute the average pooling and apply the softmax function to determine the adaptive weights $w_i, w_{ei}, w_{ee}$ for fusion:

$$(w_i, w_{ei}, w_{ee}) = \sigma(pooling((h_i, CM_{ei}, CM_{ee}))) \tag{4}$$

$$\mathcal{H}_e = w_i \odot h_i + w_{ei} \odot CM_{ei} + w_{ee} \odot CM_{ee} \tag{5}$$

where $\odot$ denotes element-wise multiplication and $\mathcal{H}_e$ is the fused enhanced representations utilized to distinguish between machine-revised and human-generated text. The cross-entropy (CE) loss function for machine-revised text detection is as follows:

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot \log \hat{y}_{i,c} \tag{6}$$

where $y_{i,c}$ denotes the one-hot encoded label of sample $i$ for class $c$ and $\hat{y}_{i,c}$ denotes the predicted probability that the sample $i$ belongs to class $c$.

### 3.2 MCL with the Length-aware Weighting (LW) Module

Supervised contrastive learning (SCL) (Gunel et al., 2021; Moukafih et al., 2022) is designed to capture the similarity among positive samples while highlighting the distinctions between positive samples and negative samples for the fine-tuning stage. In our framework, we denote the positive samples of $x_i$ as $x_i^{pos}$ and the negative samples as $x_i^{neg}$. A common approach to constructing ($x_i$,$x_i^{pos}$)pairs involves applying text augmentation techniques (e.g., word deletion and word substitution), which may inadvertently alter the inherent semantic properties of the original text. To mitigate this issue, we use RoBERTa-base as the momentum encoder and

**Algorithm 1** MCL with the LW Module

**Require:** A batch of detected texts $X_i$; Ground-truth labels $Y_i$
**Ensure:** Negative samples $X_{neg}$; Positive samples $X_{pos}$

$X_{anc} \leftarrow Main\_encoder.encode(X_i)$
$X_{pos} \leftarrow Momentum\_encoder.encode(X_i)$ ▷ Positive sample by dropout noise
$X_{pos} \leftarrow X_{pos}.detach()$ ▷ No gradient
$Momentum\_queue \leftarrow enqueue(Momentum\_queue, X_{pos})$
**if** $Momentum\_queue.shape > max\_queue\_size$ **then**
    $dequeue(Momentum\_queue)$ ▷ Dequeue the earliest batch
**end if**
▷ Negative samples
$Similarity \leftarrow sim(X_{anc}, Momentum\_queue)$
$priors \leftarrow \pi(Momentum\_queue.text\_length)$ ▷ Length-aware Weighting Module
$logits \leftarrow \sigma(Momentum\_encoder(Momentum\_queue))$
$weights \leftarrow logits \times priors$
**if** $Y_i$ **then**
    $Target\_labels \leftarrow Momentum\_queue.labels$
**else**
    $Target\_labels \leftarrow \neg Momentum\_queue.labels$
**end if**
$Negative\_weighted\_sim \leftarrow weights \times (Similarity \times (1 - Target\_labels))$ ▷ Weight the similarity
$X_{neg} \leftarrow topk(Negative\_weighted\_sim)$
**return** $X_{neg}, X_{pos}$

leverage dropout noise(Gao et al., 2021) to construct one positive sample $x_i^{pos}$ for $x_i$:

$$x_i^{mom} = Momentum\_encoder(x_i) \quad (7)$$

$$x_i^{pos} = \mathcal{E}(x_i^{mom}), \quad \mathcal{E} \triangleq dropout\ noise \quad (8)$$

To construct ($x_i$,$x_i^{neg}$)pairs, Kim et al. (2022) took all samples except for itself as $x_i^{neg}$ in a batch without leveraging label information, which hinders the model performance. Kim et al. (2024) took the samples whose labels differ from the anchor as $x_i^{neg}$. However, these strategies may pose challenges for shorter texts, as our findings reveal a high degree of similarity between extremely simple machine-revised texts and human-generated texts. Consequently, the attribution of such simple machine-revised texts remains ambiguous: while they are undeniably generated by machines, they conform closely to the patterns of human-generated texts(Tian et al., 2024).

To mitigate the impact of these false negative samples, we propose MCL with the LW module to effectively leverage text length and label information. The algorithm is outlined in Algorithm 1. Specifically, the detected texts in a batch, denoted as $X_i$, are first fed into the main encoder to extract the anchors' representations $X_{anc}$. Subsequently, we use the momentum encoder to encode the detected texts and enqueue the derived representations into the momentum queue. We then calculate the cosine similarity between $X_{anc}$ and representations of samples stored in the momentum queue, except for the anchors'. The LW module computes

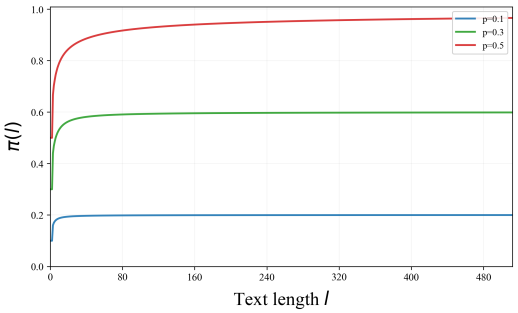



Figure 4: Variations of $\pi(l)$ with text length $l$ under different p settings.

length-aware priors based on the length of the texts in the momentum queue. Specifically, we construct a Markov chain framework to model the stochastic evolution of sequence confidence with respect to text length $l$. The state transition mechanism of the Markov chain is formulated as:

$$\sigma_{i+1} = \sigma_i P \quad (9)$$

where $\sigma_i \in \mathbb{R}^{(l+1)}$ denotes state distribution at state $i \in \{0, 1, ..., l\}$ and $P \in \mathbb{R}^{(l+1)\times(l+1)}$ denotes the state transition matrix, which is explicitly defined as:

$$P = \begin{bmatrix} p & 1-p & 0 & 0 & \cdots & 0 & 0 \\ p & 0 & 1-p & 0 & \cdots & 0 & 0 \\ 0 & p & 0 & 1-p & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1-p \\ 0 & 0 & 0 & 0 & \cdots & p & 1-p \end{bmatrix} \quad (10)$$

where $p$ denotes the negative probability of a single token. Equations 9 and 10 illustrate the state transition mechanism: a current state shifts to an adjacent state, with a negative transition probability $p$ and a non-negative transition probability $1 - p$. The final state $\sigma_l$ denotes the state distribution from initial state $\sigma_0$ after $l$ transitions:

$$\sigma_l = \sigma_0 P^l \quad (11)$$

where $\sigma_0$ denotes the one-hot initial state. The normalized expectation $\pi(l)$, serving as our length-aware prior, is defined as follows:

$$\mathbb{E}_l = \sum_{i=0}^{l} i \cdot \sigma_l(i) \quad (12)$$

$$\pi(l) = 1 - \mathbb{E}_l / l \quad (13)$$

where $\mathbb{E}_l$ denotes the expected position and we map the expected position to a value within the

interval $[0, 1]$. As shown in Figure 4, $\pi(l)$ rises rapidly with increasing text length and then gradually approaches a stable value. This value together with softmax-normalized logits from the momentum queue is then utilized to adjust the similarity scores of negative samples:

$$W^{'} = softmax(logits) \cdot \pi(L) \tag{14}$$

where $logits$ denotes the logits of all samples in the momentum queue processed by the momentum encoder. The weights $W^{'}$ align perfectly with our previous hypothesis: when the text is short, the attribution remains ambiguous, thus giving it a lower weight. We then select the negative samples from the momentum queue based on the labels.

$$W = W^{'} \cdot sim\big(X_{anc}, X^{neg}_{Mom\_q}\big) \tag{15}$$

where $X^{neg}_{Mom\_q}$ denotes the set of negative samples in the momentum queue, defined as $\{x_i \in Momentum\_queue \mid y_i \neq y_{anchor}\}$, and $sim(\cdot)$ is the cosine similarity function.

Consequently, a top-k strategy is adopted for hard negative sample mining based on the weighted similarity scores $W$:

$$X^{neg}_{N} = topk(W) \tag{16}$$

where $N$ denotes the number of selected hard negative samples from the momentum queue. The CL loss function and the total loss function for machine-revised text detection are as follows:

$$\mathcal{L}_{CL} = -\log \frac{e^{\left(sim(x_i, x_i^{pos})/\tau\right)}}{\sum_{j=1}^{N} e^{\left(sim(x_i, x_j^{neg})/\tau\right)}} \tag{17}$$

$$\mathcal{L} = (1-\lambda) \cdot \mathcal{L}_{CL} + \lambda \cdot \mathcal{L}_{CE} \tag{18}$$

where $\tau > 0$ is a scalar temperature hyperparameter and $\lambda \in (0, 1)$ is a scalar weighting hyperparameter that we tune for the detection task.

The above loss is used to update the parameters of the main encoder, whereas the parameters of the momentum encoder are updated solely via the exponential moving average (EMA) mechanism:

$$\theta_{mom} \leftarrow m \cdot \theta_{mom} + (1-m) \cdot \theta_{main} \tag{19}$$

where $\theta_{mom}$ denotes the parameters of the momentum encoder, $\theta_{main}$ denotes the parameters of the main encoder, and $m$ is a scalar hyperparameter.

# 4 Baseline

## 4.1 Experimental Datasets

We follow Nguyen-Son et al. (2024) to select 5000 samples from the XSum dataset(Narayan et al., 2018), and use different LLMs (i.e., Qwen3, Llama3, DeepSeek, GPT) to revise them as our benchmark for the homologous machine-revised text detection experiment where the training, validation, and testing subsets are derived from the same source. The prompts for revising and generating texts are in Appendix B. Our datasets cover three tasks: rewrite, polish, and generate. Specifically, rewrite and polish tasks are performed via instruction-based prompting, while generate task is implemented by sampling a text prefix and letting the model continue the sequence. These texts are then split into training, validation, and testing subsets at ratios of 80%, 10%, and 10%.

## 4.2 Baseline Models

To evaluate the effectiveness of our proposed method, we compare with the following baseline models.

- Bags-of-Words (Ippolito et al., 2020) constructs a bag-of-words embedding by counting word frequencies in the detected text, with TF-IDF weighting to quantify term importance.

- Logit-based models: Likelihood (Solaiman et al., 2019) computes the mean log probability. HistGLTRBuckets (Gehrmann et al., 2019) and Hist50Buckets (Ippolito et al., 2020) bin tokens by token probability rank across all positions. Entropy (Gehrmann et al., 2019) computes the mean entropy. DetectGPT (Mitchell et al., 2023), Fast-DetectGPT (Bao et al., 2024), and ImBD (Chen et al., 2025) are corresponding detection strategies based on the hypothesis that machines tend to generate tokens with higher statistical probability while humans demonstrate no such bias.

- Training-based models include RoBERTa-base, RoBERTa-large, SIMLLM (Nguyen-Son et al., 2024) (which leverages the similarity between an input text and its generated counterpart), MCL (Kim et al., 2024) and its variants.

We follow Bao et al. (2024) to adopt AUROC as our metric to evaluate the performance of models.

| Models | AUROC | | | | |
|---|---|---|---|---|---|
| | Qwen3 | Llama3 | DeepSeek | GPT | Avg. |
| Bags-of-Words* | 0.7709* | 0.9344* | 0.9234* | 0.8846* | 0.8783* |
| ***Logit-based Models*** | | | | | |
| HistGLTRBuckets | 0.5302 | 0.5856 | 0.5438 | 0.5665 | 0.5565 |
| Hist50Buckets | 0.4989 | 0.5448 | 0.5224 | 0.5591 | 0.5313 |
| Likelihood | 0.5342 | 0.5883 | 0.5144 | 0.5429 | 0.5449 |
| Entropy | 0.5257 | 0.5369 | 0.5091 | 0.5234 | 0.5238 |
| DetectGPT | 0.4991 | 0.4790 | 0.4345 | 0.4275 | 0.4600 |
| Fast-DetectGPT | 0.5233 | 0.5440 | 0.4475 | 0.4660 | 0.4956 |
| ImBD* | 0.8202* | 0.9346* | 0.9059* | 0.8506* | 0.8778* |
| ***Training-based Models*** | | | | | |
| RoBERTa-base | 0.4934 | 0.4917 | 0.4718 | 0.4833 | 0.4851 |
| RoBERTa-large | 0.4599 | 0.5334 | 0.5593 | 0.4990 | 0.5129 |
| SIMLLM | 0.9266 | 0.9688 | 0.9597 | 0.9430 | 0.9495 |
| **Ours*** | **0.9451(↑1.97%)*** | **0.9835(↑1.52%)*** | **0.9787(↑1.98%)*** | **0.9659(↑2.43%)*** | **0.9683(↑2.09%)*** |

Table 1: AUROC performance comparison on the Xsum LLM-rewritten dataset. The best performance across all models is in **bold**; Underlined values denote the second-best performance; and the * symbol denotes the best-performing model within each of the three categories. The percentage in parentheses represents the performance improvement of our method over the second-best model. DetectGPT leveraged T5-large (Raffel et al., 2020) for text perturbation and gpt2-medium (Radford et al., 2019) for scoring. Fast-DetectGPT leveraged falcon-7B (Technology Innovation Institute, 2023a) for sampling and falcon-7B-instruct (Technology Innovation Institute, 2023b) for scoring. ImBD leveraged gpt-neo-2.7B (Gao et al., 2020) as the reference and scoring models.

| **Models** | **AUROC** | | | |
|---|---|---|---|---|
| | rewrite | polish | generate | Avg. |
| Bags-of-Words | 0.8783 | 0.8468 | 0.8176 | 0.8476 |
| HistGLTRBuckets | 0.5565 | 0.5731 | 0.6272 | 0.5856 |
| Hist50Buckets | 0.5313 | 0.5163 | 0.5931 | 0.5469 |
| Likelihood | 0.5449 | 0.5443 | 0.6265 | 0.5719 |
| Entropy | 0.5238 | 0.5244 | 0.6097 | 0.5526 |
| DetectGPT | 0.4600 | 0.5025 | 0.5287 | 0.4971 |
| Fast-DetectGPT | 0.4956 | 0.5339 | 0.6672 | 0.5656 |
| ImBD | 0.8778 | 0.8441 | 0.8510 | 0.8576 |
| RoBERTa-base | 0.4851 | 0.4627 | 0.4732 | 0.4737 |
| RoBERTa-large | 0.5129 | 0.4424 | 0.4933 | 0.4829 |
| SIMLLM | 0.9495 | 0.9500 | 0.9497 | 0.9497 |
| **Ours** | **0.9683** | **0.9661** | **0.9567** | **0.9637** |

Table 2: Comparison of the average AUROC performance on rewrite, polish, and generate tasks.

| **Scenario** | **Models** | **Train** | **Test** | **AUROC** |
|---|---|---|---|---|
| Cross-prompt | Bags-of-Words | rewrite | polish | 0.7063 |
| | | polish | rewrite | 0.7635 |
| | ImBD | rewrite | polish | 0.7671 |
| | | polish | rewrite | 0.7685 |
| | SIMLLM | rewrite | polish | 0.9120 |
| | | polish | rewrite | 0.9195 |
| | **Ours** | rewrite | polish | **0.9285** |
| | | polish | rewrite | **0.9275** |
| Cross-LLM | Bags-of-Words | Qwen3 | DeepSeek | 0.8758 |
| | | DeepSeek | Qwen3 | 0.7390 |
| | ImBD | Qwen3 | DeepSeek | 0.8743 |
| | | DeepSeek | Qwen3 | 0.7456 |
| | SIMLLM | Qwen3 | DeepSeek | 0.9441 |
| | | DeepSeek | Qwen3 | 0.8306 |
| | **Ours** | Qwen3 | DeepSeek | **0.9663** |
| | | DeepSeek | Qwen3 | **0.8752** |

Table 3: AUROC performance comparison on cross-prompt and cross-LLM scenarios. LLM for revision on cross-prompt scenarios: Qwen3.

# 5 Results and Analysis

## 5.1 Evaluation on the Xsum LLM-rewritten dataset

To evaluate the method's effectiveness on the homogeneous dataset, we conducted experiments on the Xsum dataset that is rewritten by diverse LLMs.

As shown in Table 1, the mainstream logit-based models (e.g., DetectGPT) and pre-trained RoBERTa models exhibit extremely limited performance in detecting machine-revised text, with their average AUROC scores merely hovering around 0.46-0.56, which is comparable to random guessing. In contrast, among logit-based models, the ImBD achieves competitive performance comparable to the Bags-of-Words model, with an average AUROC of 0.8778. Notably, our method outperforms state-of-the-art SIMLLM across all evaluated LLMs. Specifically, our method achieves an average AUROC of 0.9683, with a 2.09% relative improvement over SIMLLM (0.9495).

## 5.2 Robustness to diverse scenarios

**Evaluation on polish and generate tasks.** We conducted additional experiments on polish and generate tasks using the XSum dataset. As shown in Table 2, our method achieves an average AUROC of 0.9637 and outperforms all baseline models across the three tasks. The detail of the results can be available in Appendix D.3.

**Evaluation on cross-scenario datasets.** We conducted experiments on cross-scenario XSum

| Modules | | AUROC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| LW | EBD | Qwen3 | | Llama3 | | DeepSeek | | GPT | | Avg. |
| | | rewrite | polish | rewrite | polish | rewrite | polish | rewrite | polish | |
| × | × | 0.9387 | 0.9213 | 0.9796 | 0.9855 | 0.9745 | 0.9679 | 0.9458 | 0.9640 | 0.9597 |
| ✓ | × | 0.9392 | 0.9254 | 0.9801 | 0.9839 | 0.9753 | 0.9685 | 0.9470 | 0.9639 | 0.9604 |
| × | ✓ | 0.9450 | 0.9277 | 0.9822 | 0.9861 | 0.9783 | 0.9751 | 0.9588 | 0.9654 | 0.9648 |
| ✓ | ✓ | **0.9451** | **0.9367** | **0.9835** | **0.9866** | **0.9787** | **0.9753** | **0.9659** | **0.9657** | **0.9672** |

Table 4: Ablation study results on core modules of our method.

| Models | AUROC | | | | |
|---|---|---|---|---|---|
| | XSum | Writing | PubMed | SQuAD | Avg. |
| Bags-of-Words | 0.9890 | 0.8498 | 0.7772 | 0.9227 | 0.8846 |
| HistGLTRBuckets | 0.6232 | 0.6175 | 0.5334 | 0.5499 | 0.5810 |
| Hist50Buckets | 0.5050 | 0.5335 | 0.5327 | 0.5117 | 0.5207 |
| Likelihood | 0.5734 | 0.5580 | 0.5486 | 0.5508 | 0.5577 |
| Entropy | 0.4292 | 0.6102 | 0.5986 | 0.3114 | 0.4873 |
| DetectGPT | 0.4042 | 0.6986 | 0.5362 | 0.5442 | 0.5433 |
| Fast-DetectGPT | 0.3507 | 0.6103 | 0.5213 | 0.4621 | 0.4833 |
| ImBD | 0.9085 | 0.8706 | 0.7114 | 0.8509 | 0.8353 |
| RoBERTa-base | 0.5211 | 0.6695 | 0.4450 | 0.5852 | 0.5552 |
| RoBERTa-large | 0.4419 | 0.6427 | 0.4775 | 0.5530 | 0.5287 |
| SIMLLM | 0.9911 | 0.9462 | 0.8343 | 0.9462 | 0.9318 |
| **Ours** | **0.9985** | **0.9735** | **0.8579** | **0.9724** | **0.9555** |

Table 5: Comparison of the average AUROC performance calculated by averaging AUROCs on rewrite and polish tasks for each heterogeneous testing dataset. LLM for revision: DeepSeek.

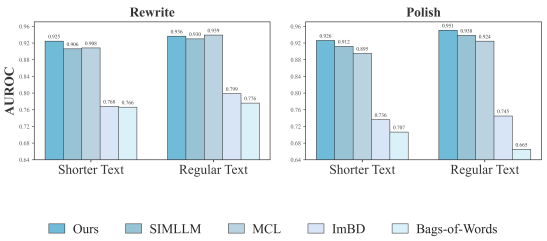



Figure 5: AUROC performance comparison on the XSum shorter and regular datasets. LLM for revision: Qwen3.

datasets. Specifically, the training and testing datasets are revised under different conditions, with either the revision prompt or the LLM used differing between the two datasets. As shown in Table 3, our method outperforms the best-performing model within each of the three categories in both cross-prompt and cross-LLM scenarios.

**Evaluation on heterogeneous datasets.** For the heterogeneous machine-revised text detection experiment where the training, validation, and testing datasets are derived from different sources, we chose the heterogeneous dataset from Chen et al. (2025) for model training. The testing datasets include XSum(Narayan et al., 2018), WritingPrompts(Fan et al., 2018), PubMedQA(Jin et al., 2019), and SQuAD(Rajpurkar et al., 2016), along with their machine-revised versions. As shown in Table 5, our method achieves an average AUROC of 0.9555 and outperforms all baseline models across the four testing datasets. The detail of the results can be available in Appendix D.4.

The results demonstrate the robustness of our method across diverse tasks, prompts, LLMs, and datasets.

### 5.3 Ablation Study

**Ablation on impact of text length.** To evaluate the effectiveness of our method in detecting short machine-revised text, we split the Qwen3-revised texts into shorter texts and regular texts on the XSum dataset using a 30-word threshold. We then conducted ablation experiments to compare our method with the best-performing model within each of the three categories, as well as the MCL baseline. As shown in Figure 5, our method demonstrates superior performance on shorter text detection. Specifically, on the Qwen3-rewritten dataset, it achieves an AUROC of 0.9246, with a 1.78% relative improvement over the MCL baseline (0.9084). On the Qwen3-polished dataset, it achieves an AUROC of 0.9262, with a 3.49% relative improvement over the MCL baseline (0.8950). These results indicate that our method exhibits robustness for shorter machine-revised text detection, while maintaining competitive performance on regular text detection.

**Ablation on core modules.** We conducted ablation experiments by removing each module of our method on the XSum dataset. As shown in Table 4, we systematically evaluate the two core modules **LW** and **EBD**. The MCL baseline achieves an average AUROC of 0.9597, demonstrating solid performance. When the LW module is introduced, the average AUROC marginally improves to 0.9604. When the EBD module is introduced, the average AUROC improves to 0.9648. The best performance is observed when both the LW module and the EBD module are introduced, with an average AUROC of 0.9672.

These results indicate that the combination of

the LW module and the EBD module can improve model performance across diverse scenarios.

## 6 Conclusion

In this paper, we propose LAMCL, a novel framework for multiscale machine-revised text detection, which utilizes the EBD module for semantic feature enhancement and MCL with the LW module for representation learning. Specifically, the EBD module fuses the input text and its LLM-processed counterpart, and then measures their semantic consistency for detection. The MCL with the LW module constructs positive samples via dropout noise, and mines hard negative samples by leveraging text length and label information, which enhances the robustness of representation learning for multiscale texts. Experimental results indicate that our method demonstrates superior AUROC performance in identifying machine-revised text across various scenarios compared to the existing detectors.

## Acknowledgments

We thank the anonymous reviewers and area chairs for their insightful comments and helpful feedback.

## Limitations

Despite the promising results, our method has several limitations. On one hand, the EBD module relies on LLMs for text processing, thus the time cost of detection is associated with the inference time of the LLMs we use. Future research should consider lightweight models or distilled small-scale models to optimize the efficiency (Tan et al., 2026). On the other hand, our work only focuses on the detection of human-generated texts and their revisions, and does not consider the scenario where humans revise machine-generated texts to evade detection, which may be a potential adversarial attack direction. Future research should focus on constructing a human-revised dataset to cover a broader range of machine-generated text detection.

## References

Philip Bachman, R. Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems 32*, Vancouver, BC, Canada.

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 *Proceedings of the 12th International Conference on Learning Representations*.

Huixin Chen, Jan Buessing, David Ruegamer, and Ercong Nie. 2024. Team mgtd4adl at semeval-2024 task 8: Leveraging (sentence) transformer models with contrastive learning for identifying machine-generated text. In *Proceedings of the 18th International Workshop on Semantic Evaluation*, pages 1711–1718.

Jiaqi Chen, Xiaoye Zhu, Tianyang Liu, Ying Chen, Chen Xinhui, Yiwen Yuan, Chak Tou Leong, Zuchao Li, Long Tang, Lei Zhang, Chenyu Yan, Guanghao Mei, Jie Zhang, and Lefei Zhang. 2025. Imitate before detect: Aligning machine stylistic preference for machine-revised text detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 23559–23567.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1575—1585.

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.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. ArXiv preprint arXiv:2101.00027.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. 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.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In *Proceedings of the 9th International Conference on Learning Representations*.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1735–1742.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. pages 9726–9735.

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.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. 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 9th International Joint Conference on Natural Language Processing*, pages 2567–2577.

Jaehoon Kim, Seungwan Jin, Sohyun Park, Someen Park, and Kyungsik Han. 2024. Label-aware hard negative sampling strategies with momentum contrastive learning for implicit hate speech detection. In *Findings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 16177–16188.

Youngwook Kim, Shinwoo Park, and Yo-Sub Han. 2022. Generalizable implicit hate speech detection using contrastive learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6667–6679.

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.

Junyu Lu, Hongfei Lin, Xiaokun Zhang, Zhaoqing Li, Tongyue Zhang, Linlin Zong, Fenglong Ma, and Bo Xu. 2023. Hate speech detection via dual contrastive learning. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 31:2787—-2795.

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.

Youness Moukafih, Abdelghani Ghanem, Karima Abidi, Nada Sbihi, Mounir Ghogho, and Kamel Smaili. 2022. Simscl: A simple fully-supervised contrastive learning framework for text representation. *Lecture Notes in Computer Science*, 13151 LNAI:728 – 738.

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.

Hoang-Quoc Nguyen-Son, Minh-Son Dao, and Koji Zettsu. 2024. Simllm: Detecting sentences generated by large language models using similarity between the generation and its re-generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22340—-22352.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21.

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*, page 2383 – 2392.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. ArXiv preprint arXiv:1908.09203.

Shilei Tan, Yongcheng Zhou, Haoxiang Liu, Xuesong Wang, Si Chen, and Wei Gong. 2026. Detinyllm: Efficient detection of machine-generated text via compact paraphrase transformation. *Information Fusion*, 127:103713.

Technology Innovation Institute. 2023a. Falcon-7b. `https://huggingface.co/tiiuae/falcon-7b`.

Technology Innovation Institute. 2023b. Falcon-7b-instruct. `https://huggingface.co/tiiuae/falcon-7b-instruct`.

Yuchuan Tian, Hanting Chen, Xutao Wang, Zheyuan Bai, Qinghua Zhang, Ruifeng Li, Chao Xu, and Yunhe Wang. 2024. Multiscale positive-unlabeled detection of ai-generated texts. In *Proceedings of the 12th International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. ArXiv preprint arXiv:2302.13971.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. 2020. Scan: Learning to classify images without labels. *Lecture Notes in Computer Science*, 12355 LNCS:268—-285.

Lingyun Xiang, Yan Li, Wei Hao, Peng Yang, and Xiaobo Shen. 2018. Reversible natural language watermarking using synonym substitution and arithmetic coding. *CMC-Computers Materials & Continua*, 55(3):541–559.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. ArXiv preprint arXiv:2505.09388.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 11613–11621.

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

Qihui Zhang, Chujie Gao, Dongping Chen, Yue Huang, Yixin Huang, Zhenyang Sun, Shilin Zhang, Weiye Li, Zhengyan Fu, Yao Wan, and Lichao Sun. 2024. Llm-as-a-coauthor: Can mixed human-written and machine-generated text be detected? In *Findings of the Association for Computational Linguistics*, pages 409–436, Hybrid, Mexico City, Mexico.

Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, and Y. X. Wei. 2025. Insights into deepseek-v3: Scaling challenges and reflections on hardware for ai architectures. ArXiv preprint arXiv:2505.09343.

## A Implementation Details

To process the detected text, we used 4 LLMs with the predefined prompts in the EBD module, as shown in Table 6. The temperature of the LLMs was set to 0.7 for text revision to balance the diversity and coherence of the generated content, and the number of heads of the MHA in the CM module was set to 8. We then fine-tuned the MCL with the following parameters: we set the batch size to 16, the max queue size to 512, the number of selected hard negative samples $N$ to 16, the momentum hyperparameter $m$ to 0.999, the CL loss temperature $\tau$ to 0.05, the loss weighting hyperparameter $\lambda$ to 0.1, the learning rate to 2e-5, and the training epoch to 10. In the LW module, the negative transition probability $p$ is typically a small value; thus, it was set to 0.1. All experiments were conducted on an NVIDIA A100 GPU (80GB).

| LLM | Version |
|---|---|
| Qwen3(Yang et al., 2025) | Qwen3-32B |
| Llama3(Touvron et al., 2023) | Meta-Llama-3-8B-Instruct |
| DeepSeek(Zhao et al., 2025) | DeepSeek-V3.2 |
| GPT(Radford and Narasimhan, 2018) | GPT-4o |

Table 6: LLMs for text revision and generation.

## B Predefined Prompts

We constructed the machine-generated dataset by using LLMs with the predefined prompts, and also use them to process the input text in the EBD module. The prompts for text rewriting, polishing and generation are as follows:

- **Polish prompt:**
  You are a professional text polisher. Please polish the given text, keeping the original meaning completely. Please only output the polished text without any explanation. Polish:$< text >$.

- **Rewrite prompt:**
  You are a professional text rewriter. Please rewrite the given text, keeping the original meaning completely. Please only output the rewritten text without any explanation. Rewrite:$< text >$.

The $< text >$ denotes the input text. It could be like "The full cost of damage in Newton Stewart, one of the areas worst affected, is still being assessed.".

- **Generate prompt:**
  You are a professional news writer. Please start your writing with the following text. The total word count (including the provided text) does not exceed 50 words. Please only output the complete text (the original provided text and your continued writing) without any explanation. Here is the provided text: $< prefix >$. Here is the complete text:.

| Bart Score | Qwen3 | | Llama3 | | DeepSeek | | GPT | |
|---|---|---|---|---|---|---|---|---|
| | rewrite | polish | rewrite | polish | rewrite | polish | rewrite | polish |
| Mean | -1.6080 | -1.1784 | -2.3264 | -2.1981 | -2.3172 | -1.8710 | -1.9117 | -1.6348 |
| Var | 0.5757 | 0.5073 | 0.6237 | 0.6235 | 0.6387 | 0.6073 | 0.6149 | 0.5956 |

Table 7: The Bart Score between human-generated texts and their revisions by LLMs.

| Bart Score | Qwen3 | | Llama3 | | DeepSeek | | GPT | |
|---|---|---|---|---|---|---|---|---|
| | rewrite | polish | rewrite | polish | rewrite | polish | rewrite | polish |
| Mean | -1.3380 | -0.7632 | -2.0663 | -1.8463 | -1.9966 | -1.4284 | -1.6155 | -1.2412 |
| Var | 0.5137 | 0.4033 | 0.6240 | 0.6085 | 0.6023 | 0.5692 | 0.5764 | 0.5254 |

Table 8: The Bart Score between machine-generated texts and their revisions by LLMs.

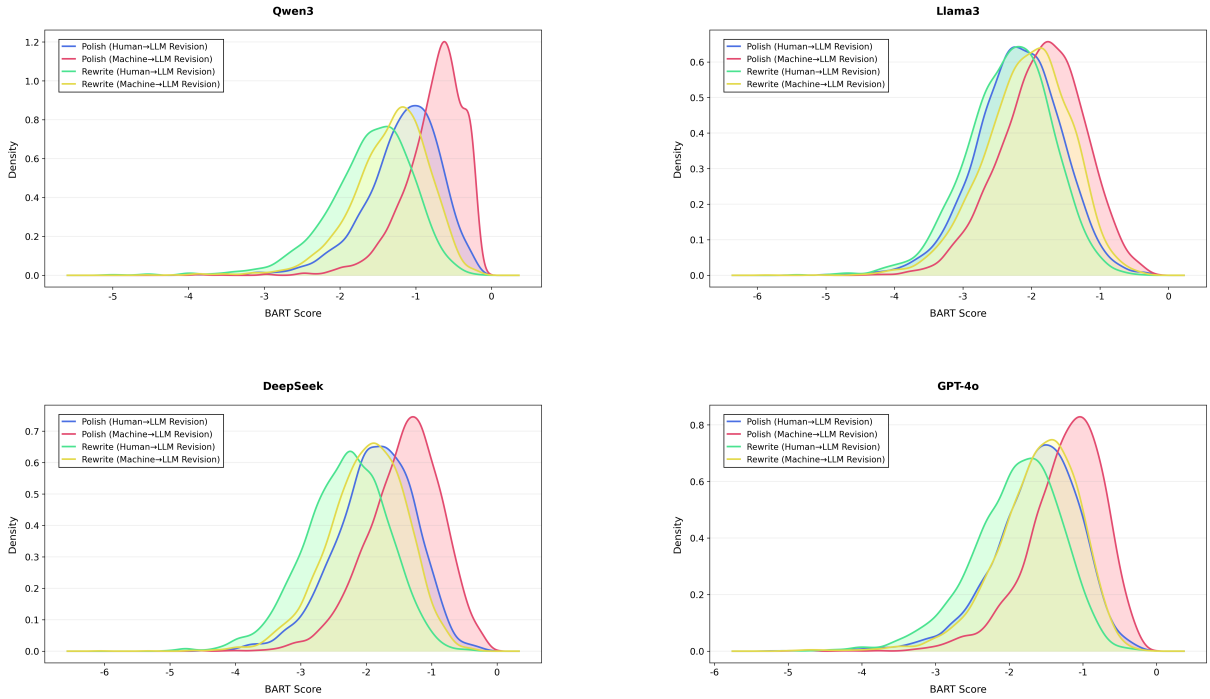



Figure 6: Comparison of BART scores between human-generated texts and machine-generated texts after LLM revision. We use 5000 samples from the XSum dataset on rewrite and polish tasks.

The $< prefix >$ denotes the first 10 words of the input text. It could be like "Repair work is ongoing in Hawick and many roads in ".

## C Discussion of Our Proposed Modules

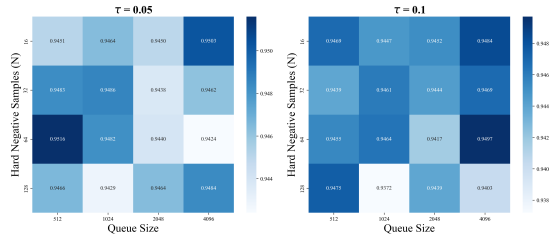
In Section 5.3, we conducted the ablation study that evaluates the effectiveness of our proposed modules. This section further explores the intrinsic working mechanism.

### C.1 Similarity assessment of the LLM-processed text

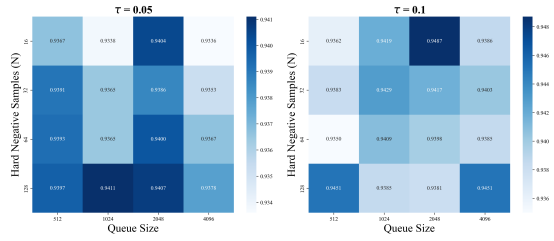
To evaluate the similarity between the original text and the processed text by LLMs, we utilized the BART score(Yuan et al., 2021), as it enables the extraction of comprehensive contextual features. And the BART score is positively correlated with text semantic similarity. As shown in Figure 6, for human-generated texts, the average BART scores corresponding to their Qwen3-rewritten and Qwen3-polished revisions are -1.608 and -1.178 respectively; for machine-generated texts, the average BART scores corresponding to their Qwen3-rewritten and Qwen3-polished revisions are -1.338 and -0.763 respectively. Notably, human-generated texts generally exhibit lower similarity with their revisions compared to machine-generated texts, which indicates the EBD module can leverage such discriminative differences to effectively distinguish between human-generated text and machine-revised text.

### C.2 Effect of hyperparameters

In the MCL framework, the max queue size and the number of selected hard negative samples $N$ determine the candidate pool and negative sample size for CL. Thus, we evaluate the effect of the two hyperparameters under different CL loss temperature $\tau$. As shown in Figure 7, a larger queue size and a larger hard negative sample size $N$ do not always improve the performance. Despite the expanding range of optional negative samples, some mismatched samples may be included in the hard negative samples, which can compromise the performance. And we further evaluate the effect of the hyperparameter negative transition probability $p$ in the LW module. As shown in Table 9, compared to the baseline (Ours without the LW module), the performance can be improved by properly setting the value of $p$. The results indicate that the LW module can utilize text length information to effectively select hard negative samples for CL.




(a) Rewrite




(b) Polish

Figure 7: AUROC performance on rewrite and polish tasks across hyperparameter configurations: CL loss temperature $\tau$, max queue size, and hard negative sample size. LLM for revision: Qwen3.

### C.3 Ablation on connection strategy in the EBD module.

In the EBD module, we utilize the CM module to measure the semantic consistency between the input text and its LLM-processed counterpart. To validate the necessity of the CM module, we further conduct ablation experiments by comparing it with a baseline that directly feeds the concatenated enhanced text representations into the head layer without the CM module. As shown in Table 10, our method achieves an average AUROC of 0.9672, outperforming the baseline (0.9658). The results indicate that the CM Module can improve the model performance by measuring the semantic consistency.

| $p$ | AUROC | |
|---|---|---|
| | rewrite | polish |
| baseline | 0.9450 | 0.9277 |
| 0.1 | 0.9451 | 0.9367 |
| 0.2 | **0.9467** | 0.9384 |
| 0.3 | 0.9462 | 0.9388 |
| 0.4 | 0.9442 | **0.9412** |
| 0.5 | 0.9444 | 0.9380 |

Table 9: AUROC performance on rewrite and polish tasks with varying negative transition probability $p$ in the LW module. Performance of the baseline (Ours without LW module) is included for comparison. LLM for revision: Qwen3.

## D Additional Results

### D.1 Additional evaluation on cross-scenario datasets

In Section 5.2, we conducted cross-prompt and cross-LLM experiments on the XSum dataset. We further evaluate the model's robustness via additional cross-generalization experiments. As shown in Table 11, our method outperforms the best-performing model within each of the three categories in the cross-prompt and cross-LLM generalization experiments.

### D.2 Runtime Analysis

As discussed in Limitations, the time cost of detection is associated with the inference time of the LLMs we use. We conducted efficiency experiments on the XSum LLM-rewritten dataset to evaluate the average inference time per sample. To further validate the feasibility of optimizing efficiency, we fine-tuned a lightweight BART model using original texts and their corresponding LLM-generated counterparts, and replaced the large LLM in the EBD module with this distilled small-scale model to produce the processed text counterparts. As shown in Table 12, our lightweight BART-based variant achieves a substantial improvement in inference efficiency, reducing the generation time per sample from 0.8400s to 0.1910s, while maintaining a competitive AUROC performance. This confirms that our framework can be effectively adapted to lightweight models for real-world deployment, balancing detection accuracy and inference efficiency.

| Connection Strategy (EBD) | AUROC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Qwen3 | | Llama3 | | DeepSeek | | GPT | | Avg. |
| | rewrite | polish | rewrite | polish | rewrite | polish | rewrite | polish | |
| Direct Connection | 0.9442 | 0.9366 | 0.9826 | 0.9864 | 0.9786 | 0.9688 | 0.9635 | 0.9654 | 0.9658 |
| by the CM Module | **0.9451** | **0.9367** | **0.9835** | **0.9866** | **0.9787** | **0.9753** | **0.9659** | **0.9657** | **0.9672** |

Table 10: Ablation study results on connection strategy in the EBD module.

| Scenario | Models | Train | Test | AUROC |
|---|---|---|---|---|
| Cross-Prompt and Cross-LLM | Bags-of-Words | rewrite(Qwen3) | polish(DeepSeek) | 0.8340 |
| | | polish(DeepSeek) | rewrite(Qwen3) | 0.7587 |
| | ImBD | rewrite(Qwen3) | polish(DeepSeek) | 0.8647 |
| | | polish(DeepSeek) | rewrite(Qwen3) | 0.7392 |
| | SIMLLM | rewrite(Qwen3) | polish(DeepSeek) | 0.9673 |
| | | polish(DeepSeek) | rewrite(Qwen3) | 0.9110 |
| | **Ours** | rewrite(Qwen3) | polish(DeepSeek) | **0.9802** |
| | | polish(DeepSeek) | rewrite(Qwen3) | **0.9457** |

Table 11: AUROC performance comparison on cross-prompt and cross-LLM scenarios.

| Models | Generate(s) | Detect(s) | Total(s) | AUROC |
|---|---|---|---|---|
| Ours with BART (w/o LLM) | **0.1910** | 0.0071 | 0.1981 | 0.9617 |
| Ours | 0.8400 | 0.0071 | 0.8471 | **0.9683** |
| Bags-of-Words | 0 | **0.0011** | **0.0011** | 0.8783 |
| HistGLTRBuckets | 0 | 0.0207 | 0.0207 | 0.5565 |
| Hist50Buckets | 0 | 0.0207 | 0.0207 | 0.5313 |
| Likelihood | 0 | 0.0233 | 0.0233 | 0.5449 |
| Entropy | 0 | 0.0214 | 0.0214 | 0.5238 |
| DetectGPT | 0 | 3.8823 | 3.8823 | 0.4600 |
| Fast-DetectGPT | 0 | 0.0344 | 0.0344 | 0.4956 |
| ImBD | 0 | 0.0362 | 0.0362 | 0.8778 |
| RoBERTa-base | 0 | 0.0064 | 0.0064 | 0.4851 |
| RoBERTa-large | 0 | 0.0194 | 0.0194 | 0.5129 |
| SIMLLM | 0.8400 | 0.0064 | 0.8464 | 0.9495 |

Table 12: Comparison of the average time cost per sample and AUROC performance on the XSum LLM-rewritten dataset.

### D.3 Detail of the results on polish and generate tasks

As shown in Table 13 and Table 14, logit-based models consistently perform better on generate task than on rewrite and polish tasks. And the ImBD achieves competitive performance comparable to the Bags-of-Words model, reaching an average AUROC of 0.8441 on the LLM-polished dataset, and 0.8510 on the LLM-generated dataset. Notably, our method outperforms state-of-the-art SIMLLM across all evaluated LLMs on both tasks. Specifically, our method achieves an average AUROC of 0.9661 on the LLM-polished dataset, with a 1.69% relative improvement over SIMLLM (0.9500); on the LLM-generated dataset, our method achieves an average AUROC of 0.9567, with a 0.74% relative improvement over SIMLLM (0.9497).

### D.4 Detail of the results on the heterogeneous datasets

We chose the dataset provided in the (Chen et al., 2025) paper as our benchmark for the heterogeneous machine-revised text detection experiment where the training, validation, and testing datasets are derived from different sources. Specifically, the human-generated texts are crawled from the internet before 2019, which are then revised by DeepSeek as the machine-revised texts. The dataset comprises 1,000 samples in total, covering a diverse range of domains (i.e., papers, blogs, letters, emails, homework), which are used for training. And the testing datasets include XSum(Narayan et al., 2018), WritingPrompts(Fan et al., 2018), PubMedQA(Jin et al., 2019), and SQuAD(Rajpurkar et al., 2016), along with their machine-revised versions.

As shown in Table 15, the mainstream logit-based models (e.g., DetectGPT) and pre-trained Roberta models exhibit extremely limited performance in detecting machine-revised text, with their average AUROC scores merely hovering around 0.48–0.58, which is comparable to random guessing. In contrast, among logit-based models, the ImBD achieves competitive performance comparable to the Bags-of-Words model, with an average AUROC of 0.8353. Notably, our method outperforms the state-of-the-art SIMLLM across all evaluated LLMs. Specifically, our method achieves an average AUROC of 0.9555, with a 2.54% relative improvement over SIMLLM (0.9318).

| Models | AUROC | | | | |
|---|---|---|---|---|---|
| | Qwen3 | Llama3 | DeepSeek | GPT | Avg. |
| Bags-of-Words* | 0.7284* | 0.9274* | 0.8783* | 0.8529* | 0.8468* |
| ***Logit-based Models*** | | | | | |
| HistGLTRBuckets | 0.5418 | 0.6400 | 0.5626 | 0.5481 | 0.5731 |
| Hist50Buckets | 0.5080 | 0.5673 | 0.4941 | 0.4958 | 0.5163 |
| Likelihood | 0.5116 | 0.6328 | 0.5163 | 0.5166 | 0.5443 |
| Entropy | 0.5221 | 0.5634 | 0.5062 | 0.5057 | 0.5244 |
| DetectGPT | 0.5258 | 0.5124 | 0.4850 | 0.4869 | 0.5025 |
| Fast-DetectGPT | 0.5440 | 0.5863 | 0.4940 | 0.5112 | 0.5339 |
| ImBD* | 0.7663* | 0.9394* | 0.8586* | 0.8121* | 0.8441* |
| ***Training-based Models*** | | | | | |
| RoBERTa-base | 0.5225 | 0.4205 | 0.4553 | 0.4525 | 0.4627 |
| RoBERTa-large | 0.4408 | 0.4155 | 0.4259 | 0.4873 | 0.4424 |
| SIMLLM | 0.9128 | 0.9681 | 0.9643 | 0.9547 | 0.9500 |
| **Ours*** | **0.9367(↑2.62%)*** | **0.9866(↑1.91%)*** | **0.9753(↑1.14%)*** | **0.9657(↑1.15%)*** | **0.9661(↑1.69%)*** |

Table 13: AUROC performance comparison on the LLM-polished dataset.

| Models | AUROC | | | | |
|---|---|---|---|---|---|
| | Qwen3 | Llama3 | DeepSeek | GPT | Avg. |
| Bags-of-Words* | 0.7451* | 0.8936* | 0.7558* | 0.8757* | 0.8176* |
| ***Logit-based Models*** | | | | | |
| HistGLTRBuckets | 0.5604 | 0.8265 | 0.5471 | 0.5748 | 0.6272 |
| Hist50Buckets | 0.5297 | 0.7828 | 0.5376 | 0.5222 | 0.5931 |
| Likelihood | 0.5770 | 0.8304 | 0.5420 | 0.5566 | 0.6265 |
| Entropy | 0.5952 | 0.6406 | 0.6269 | 0.5761 | 0.6097 |
| DetectGPT | 0.5161 | 0.6068 | 0.5287 | 0.4631 | 0.5287 |
| Fast-DetectGPT | 0.6141 | 0.8809 | 0.5487 | 0.6251 | 0.6672 |
| ImBD* | 0.7749* | 0.8654* | 0.8231* | 0.9406* | 0.8510* |
| ***Training-based Models*** | | | | | |
| RoBERTa-base | 0.5046 | 0.4683 | 0.4528 | 0.4672 | 0.4732 |
| RoBERTa-large | 0.5107 | 0.3626 | 0.5296 | 0.5701 | 0.4933 |
| SIMLLM | 0.9435 | 0.9547 | 0.9311 | 0.9695 | 0.9497 |
| **Ours*** | **0.9460(↑0.26%)*** | **0.9706(↑1.67%)*** | **0.9396(↑0.91%)*** | **0.9704(↑0.09%)*** | **0.9567(↑0.74%)*** |

Table 14: AUROC performance comparison on the LLM-generated dataset.

| Models | AUROC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | XSum | | Writing | | PubMed | | SQuAD | | Avg. |
| | rewrite | polish | rewrite | polish | rewrite | polish | rewrite | polish | |
| Bags-of-Words* | 0.9918* | 0.9862* | 0.9005* | 0.7990* | 0.7920* | 0.7623* | 0.9388* | 0.9065* | 0.8846* |
| ***Logit-based Models*** | | | | | | | | | |
| HistGLTRBuckets | 0.6530 | 0.5933 | 0.5136 | 0.7213 | 0.4531 | 0.6137 | 0.6481 | 0.4518 | 0.5810 |
| Hist50Buckets | 0.5198 | 0.4902 | 0.5505 | 0.5164 | 0.5094 | 0.5560 | 0.5366 | 0.4868 | 0.5207 |
| Likelihood | 0.5967 | 0.5501 | 0.3833 | 0.7326 | 0.4711 | 0.6261 | 0.5938 | 0.5077 | 0.5577 |
| Entropy | 0.3893 | 0.4690 | 0.5763 | 0.6440 | 0.5626 | 0.6346 | 0.2928 | 0.3299 | 0.4873 |
| DetectGPT | 0.3419 | 0.4665 | 0.6626 | 0.7346 | 0.4810 | 0.5913 | 0.4913 | 0.5971 | 0.5433 |
| Fast-DetectGPT | 0.2799 | 0.4215 | 0.5873 | 0.6333 | 0.4499 | 0.5926 | 0.3975 | 0.5267 | 0.4833 |
| ImBD* | 0.9307* | 0.8862* | 0.9019* | 0.8393* | 0.7110* | 0.7118* | 0.8707* | 0.8311* | 0.8353* |
| ***Training-based Models*** | | | | | | | | | |
| RoBERTa-base | 0.4801 | 0.5620 | 0.8224 | 0.5165 | 0.4772 | 0.4127 | 0.6958 | 0.4746 | 0.5552 |
| RoBERTa-large | 0.3510 | 0.5328 | 0.6904 | 0.5949 | 0.5694 | 0.3856 | 0.5192 | 0.5867 | 0.5287 |
| SIMLLM | 0.9921 | 0.9901 | 0.9688 | 0.9236 | 0.8155 | 0.8531 | 0.9423 | 0.9501 | 0.9318 |
| **Ours*** | **0.9996*** | **0.9973*** | **0.9910*** | **0.9560*** | **0.8402*** | **0.8755*** | **0.9744*** | **0.9703*** | **0.9555(↑2.54%)*** |

Table 15: AUROC performance comparison across heterogeneous datasets on rewrite and polish tasks. LLM for revision: DeepSeek.