https://doi.org/10.1007/sxxxxx-yyy-zzzz-1

RESEARCH ARTICLE

PhantomHunter: Detecting Unseen Privately-Tuned LLM-Generated Text via Family-Aware Learning

Yehan YANG[1,2,*], Yuhui SHI[1,2,*], Qiang SHENG[1,2], Hao MI[1,2], Beizhe HU[1,2], Chaoxi XU[1,2], Juan CAO[1,2]✉

1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
2. University of Chinese Academy of Sciences, Beijing 100049, China

Received month dd, yyyy; accepted month dd, yyyy

E-mail: caojuan@ict.ac.cn. * These authors contributed equally to this work.

Abstract

With the popularity of large language models (LLMs), undesirable societal problems like misinformation production and academic misconduct have been more severe, making LLM-generated text detection of unprecedented importance. Though existing methods have made remarkable progress, they mostly consider publicly known LLMs for testing the performance, and the new challenge brought by text from privately-tuned LLMs is largely underexplored. Due to the rapid development of open-source models like the Qwen and LLaMA series, even ordinary users can now easily possess private LLMs by further tuning an open-source one with private corpora. This could lead to a significant performance drop of existing detectors (by 41% at most in our preliminary study), due to their poor capability of capturing the essential LLM traits robust to fine-tuning operations. To address this issue, we propose PhantomHunter, an LLM-generated text detector specialized for detecting text from unseen privately-tuned LLMs. Instead of memorizing individual characteristics, PhantomHunter's family-aware learning framework captures family-level traits shared across the base models and their derivatives. It first extracts base model features and distills them into a lightweight proxy module for deployment efficiency consideration, followed by a contrastive family-aware learning module that enhances the family-shared information. The enhanced features are then fed into a mixture-of-experts module containing multiple experts for corresponding families for final predictions. Experiments on data from four widely-adopted LLM families (LLaMA, Gemma, Mistral, and Qwen) show PhantomHunter's superiority over 9 baselines and 11 industrial services. Our code is available at https://github.com/ICTMCG/Phantomhunter.

Key words

LLM-generated Text Detection, Family-Aware Learning

1 Introduction

Large language models (LLMs) have successfully revolutionized the way people organize and produce text and inspire productivity in a wide range of applications [1]. However, undesirable societal problems caused by the misuse of LLMs also rapidly emerge, such as cheating in academic writing [2], creating misinformation [3], and accelerating information system pollution [4]. As the front-line barrier against such threats, automatic detection techniques of LLM-generated text (LLMGT) are now of unprecedented importance and attract researchers from both academia and industry.

LLMGT detection generally distinguishes human-written and LLM-generated text via binary classification. Existing methods either learn common textual features (e.g., stylistic clues [5]) shared across LLMs using representation learning or design distinguishable metrics between human and LLM texts based on LLMs' internal signals (e.g., token probabilities [6]). For both categories, their tests were mostly conducted on data from publicly available LLMs, assuming that users generate text using public, off-the-shelf services. **We argue that this situation is being changed due to the recent development of the open-source LLM community.** With the help of platforms like HuggingFace[1] and efficient LLM training techniques like low-rank adaptation (LoRA) [7], building fine-tuned LLMs with customized private datasets has become much easier than before. For instance, there have been over 200k Qwen-based and 85k Llama-based derivative models on HuggingFace [8,9]. As shown in figure 1, after private fine-tuning on unknown corpora, the learned characteristics of base models could change, and the LLMGT detectors would fail (will empirically show in the following section), shaping a new risk that malicious users can generate harmful texts privately without being caught by LLMGT detectors. A new challenge arises: **How could we detect text generated by privately-tuned open-source LLMs?**

To address this issue, we first conduct an analysis of differences between the base LLMs and their derivative models via fine-tuning on data of different topics. The result reveals that, compared to other models, the token probability lists for text from the fine-tuned LLM are more similar to those from the base LLM, exhibiting clear "family traits." Inspired by this finding, we propose to perform family-

[1] https://huggingface.co/models

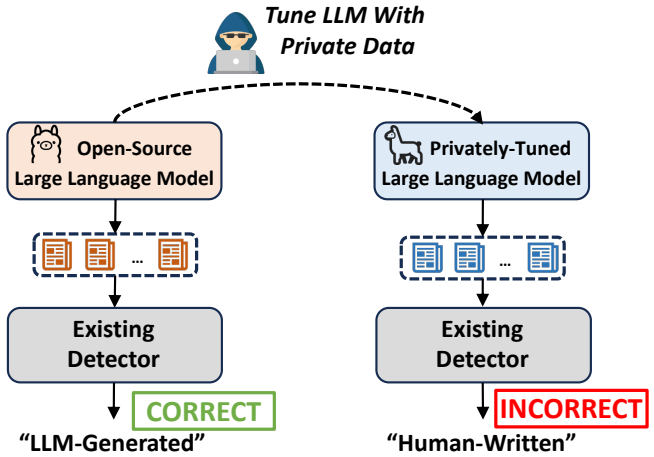

Fig. 1 Illustration of how private fine-tuning LLMs affects LLM-generated text detection. Existing detectors may successfully identify text generated by the original open-source LLM, but fail to detect text generated by the privately fine-tuned one due to its reliance on non-shared features among the base and fine-tuned LLMs, causing incorrect detection.

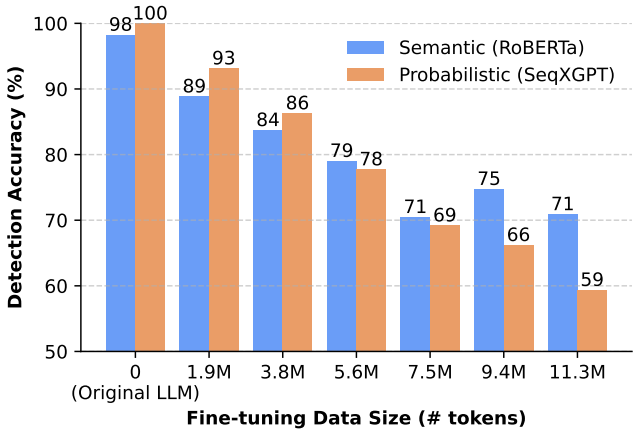

Fig. 2 Detection accuracy for LLM-generated text with increasing amounts of fine-tuning data.

aware learning by extracting probabilistic features of base models and design **PhantomHunter**, a detector targeted at text generated by unseen, privately-tuned LLMs. PhantomHunter consists of three main components: First, it obtains probability features from widely known base LLMs such as Qwen and LLaMA, and then distills them into a lightweight proxy module to improve deployment efficiency. The features are then enhanced using a contrastive family-aware learning to better preserve the family-level traits shared across the text pieces from both the base model and the fine-tuned derivative models. The enhanced features are fed into a Mixture-of-Experts (MoE) module that contains multiple expert modules specialized in detecting texts from specific LLM families. Finally, the collective judgment of the given text being LLM-generated or not is made by the gate mechanism controlled by the family prediction results. Experiments against baseline methods from both previous literature and the real-world industrial services demonstrate that our proposed PhantomHunter could better capture family-level traits and effectively detect generated text from unseen privately-tuned LLMs.

Our main contributions are summarized as follows:

- **New Risk:** We empirically expose the new risk that current LLM-generated text detectors may fail to detect text from privately-tuned open-source LLMs, even having been trained on text from the base version.
- **New Method:** We propose PhantomHunter, which is trained via family-aware learning to learn common traits shared in typical open-source LLM families and can identify text generated by privately-tuned LLMs.
- **High Performance:** We construct a new dataset containing texts in academic abstracts and Q&A genres generated by humans and four widely-adopted LLM families (LLaMA, Gemma, Mistral, and Qwen) for evaluation. Experiments show that PhantomHunter outperforms 9 baselines and 11 industrial services.

2 Preliminaries: How Does Fine-tuning Affect LLM-Generated Text Detection?

2.1 Impact of Fine-tuning on Detectability

To explore the extent to which fine-tuning affects the performance of LLM-generated text detectors, we fine-tuned LLaMA-2 7B using the arXiv abstract corpus and saved checkpoints at different fine-tuning steps, which were then used to generate test samples. We experimented on two detectors, RoBERTa (semantic-based; [5]) and SeqXGPT (probability-based; [10]). Figure 2 displays **their decaying trends in accuracy as the fine-tuning corpus size increases**. For the LLM fine-tuned on 11.3M tokens, SeqXGPT, which performs almost *perfect* at detecting text pieces from the original LLM, even encounters an astonishing accuracy drop of 41%.

2.2 Impact of Fine-tuning on Features

Experimental Settings. We fine-tuned four open-source LLMs, including LLaMA, Gemma, Mistral, and Qwen, using two sets of corpora from different domains (computer science and physics, see the Dataset Construction section for details) and collected responses from 12 model variants in total: 4 base models and 8 LoRA fine-tuned models (2 domains × 4 base models). All responses were generated under identical decoding configurations (here, temperature = 1.0, top-p = 0.95) to ensure consistency.

Method. We implement a cross-model behavioral consistency analysis to compute a relative negative log-likelihood (NLL) distance. For a given source model S and target model T, we extract their respective NLL sequences $\text{NLL}_S = [\ell_1^S, \ell_2^S, \cdots, \ell_n^S]$ and $\text{NLL}_T = [\ell_1^T, \ell_2^T, \cdots, \ell_n^T]$ over the same input sequence, where $\ell_i = -\log p_i$ represents the negative log-likelihood loss of the i-th token with predicted probability p_i.The smaller ℓ value indicates higher confidence and higher probability for the corresponding token. Therefore, to measure the overall consistency between their predictive behaviors, we compute the normalized similarity score based on the complement of

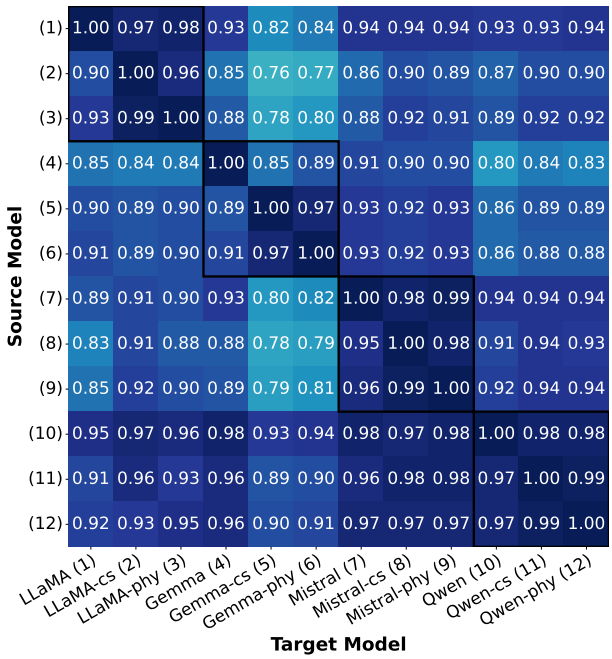

Fig. 3 Heatmap of cross-model similarity measured by the complement of relative NLL distance. Rows indicate source models and columns indicate target models, covering 12 variants across four model families and two training domains. Darker colors indicate higher similarity.

relative NLL distance:

$$\mathrm{dist}(\mathrm{NLL}_S, \mathrm{NLL}_T) = 1 - \frac{\left|\sum_{i=1}^{n} \ell_i^T - \sum_{i=1}^{n} \ell_i^S\right|}{\sum_{i=1}^{n} \ell_i^S}, \tag{1}$$

We aggregated these scores across all test samples to obtain the average similarity for each source-target model pair. The resulting 12×12 similarity matrix was visualized as a heatmap (Figure 3), with color intensity ranging from 0.5 to 1.0 to highlight the similarity in cross-model prediction patterns, providing insights into which model combinations exhibit stronger behavioral alignment.

Findings and Inspiration. From Figure 3, we intuitively observe that the probability lists of LLMs from the same family are more similar to each other, and different from those of the others, which shows significant "family traits." Despite exceptions, an overall trend of family-wise LLM similarity persists. Exceptions may be caused by the characteristics of different families and the coarse-grained metric. This observation reveals that the probability lists of the base models could be useful for modeling unseen fine-tuned ones. However, such "family traits" are not fully exploited by existing detectors because they treat all LLMs equally and may only learn the commonality across all involved LLMs (regarding the 12×12 area in the heatmap), not the family-level commonality (regarding every 3×3 areas), shaping a looser boundary against human text. Targetedly, we design PhantomHunter to learn such traits to address this issue.

3 Proposed Method: PhantomHunter

PhantomHunter is a family-aware learning-based detector for unseen, privately-tuned LLM-generated texts. The key insight behind PhantomHunter is to model the inherent family-level features shared between base LLMs and their fine-tuned descendants. Building on these observations, PhantomHunter employs a family-aware learning approach to detect texts from unseen, privately-tuned LLMs. As illustrated in Figure 4, the architecture consists of three main components: **1)** A base probability feature extractor, **2)** a contrastive learning-based family encoder, and **3)** a mixture-of-experts detection module, which will be detailed below.

3.1 Base Probability Feature Extraction

Feature Extraction. For a given text sequence $\mathbf{x}$, we pass it through M base LLMs $\{\theta_1, \theta_2, \cdots, \theta_M\}$ to obtain the probability lists $\mathbf{p}_M \in \mathbb{R}^{M\times N}$:

$$\mathbf{p}_M = (\mathbf{p}^{\theta_1}, \mathbf{p}^{\theta_2}, \cdots, \mathbf{p}^{\theta_M}), \tag{2}$$

where $\mathbf{p}^{\theta_i} = (p_1^{\theta_i}, p_2^{\theta_i}, \cdots, p_N^{\theta_i})$, N is the text sequence length, M is the number of base models (*i.e.,* families), and $p_j^{\theta_i}$ represents the probability of generating token x_j given context $x_{<j}$ using LLM θ_i.

Probability List Alignment. To address the discrepancy in the lengths of token-wise probability sequences caused by the different tokenizers of the base models in PhantomHunter, we adopt the alignment strategy proposed by Sniffer [11] and SeqXGPT [10]. Specifically, we implement a sampling and truncation strategy that utilizes "word" as unified anchors. First, the model identifies the common semantic starting point of the sequences using the initial index lists from each tokenizer, thereby eliminating head-side variances caused by prompt offsets. Subsequently, token-level features of varying granularities are aligned to the original word-flow space of the text. Through a two-stage process involving local maximum length calculation within each batch and global padding/truncation to a predefined maximum length (max_len), the non-uniform probability distributions from multiple models are aggregated into a unified feature matrix $L = [l_1, \ldots, l_t]$. In this matrix, each word-level feature vector l_i consolidates the log-likelihood characteristics of different models for the same semantic unit, achieving a rigorous word-wise alignment for multi-task modeling.

Lightweight Proxy Module Learning. A key bottleneck in deploying the proposed detection framework is that obtaining token-level probability lists requires forward passes through every base LLM at inference time, incurring computational overhead proportional to the number of base models. To eliminate this dependency, we introduce a lightweight proxy module P_ϕ that learns to approximate the base probability list directly from the input text. Concretely, P_ϕ consists of a frozen RoBERTa encoder and a two-layer MLP. The frozen encoder first maps the input text into hidden states $\mathbf{H} \in \mathbb{R}^{B\times N\times d_r}$, where B denotes the batch size, N denotes the sequence length, and d_r denotes the hidden dimension of RoBERTa. The MLP then predicts the proxy probability list $\hat{\mathbf{P}}_M \in \mathbb{R}^{B\times M\times N}$, imitating the base probability list $\mathbf{P}_M \in \mathbb{R}^{B\times M\times N}$ extracted from the M base LLMs. Each element in $\mathbf{P}_M$ or $\hat{\mathbf{P}}_M$ corresponds to the token-level probability feature.

The proxy module is trained jointly with the downstream detector through an MSE distillation loss:

$$\mathcal{L}_{\text{proxy}} = \frac{1}{B \cdot M \cdot N} \left\|\hat{\mathbf{P}}_M - \mathbf{P}_M\right\|_2^2. \tag{3}$$

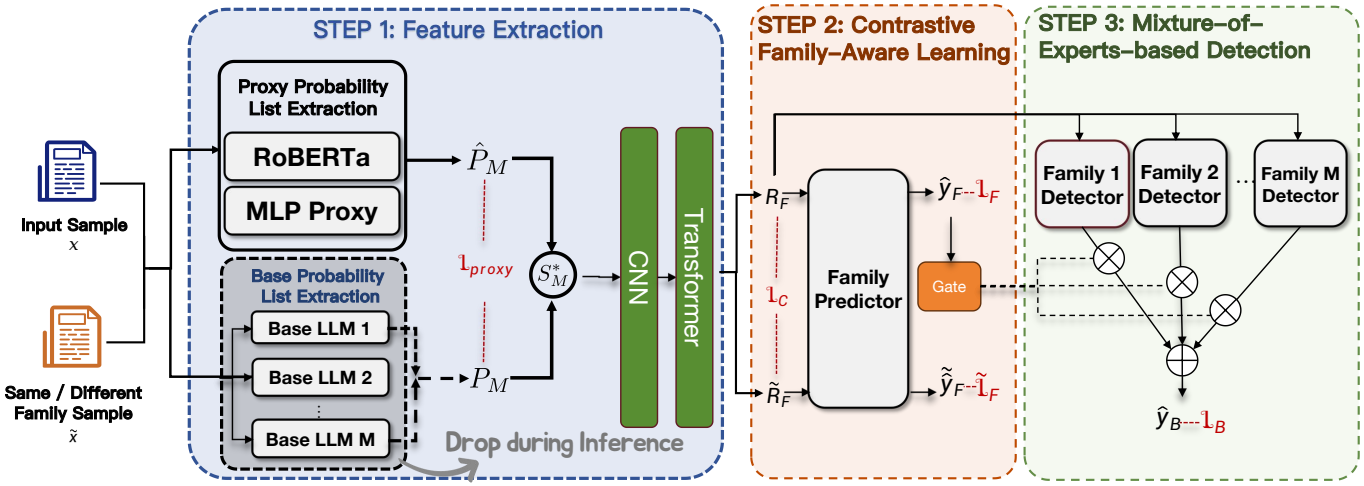

Fig. 4 Overall architecture of **PhantomHunter** and the training process. Given a text sample **x**, it **1)** extracts the probability feature from M base models and encode them with CNN and transformer blocks; **2)** predicts the family of **x** to determine the family gating weights; and **3)** feeds the representation $\mathbf{R}_F$ to a mixture-of-experts network controlled by the gating weights from Step 2 for final prediction of **x** being LLM-generated. During training, contrastive learning is applied in each mini-batch to better model family relationships. The red terms are loss functions.

where $B \cdot M \cdot N$ normalizes the squared error over all samples, base LLMs, and token positions. This objective distills the probability list of the M LLMs into a lightweight proxy. A challenge in proxy-based training is the train-inference mismatch if the detector is trained exclusively on $\mathbf{P}_M$ but relies only on $\hat{\mathbf{P}}_M$ at test time, the distributional shift between the two feature sources may degrade performance. To address this, we employ a scheduled sampling strategy [12] that stochastically combines the base and proxy probability lists during training. Specifically, at epoch t, the framework constructs the feature input $\mathbf{S}^*_M$ as:

$$\mathbf{S}^*_M = (1 - z_t)\mathbf{P}_M + z_t\hat{\mathbf{P}}_M, \quad z_t \sim \text{Bernoulli}(p_t), \tag{4}$$

where p_t is the probability of using the proxy prediction. We further adopt a curriculum learning schedule [13] in which both the sampling probability p_t and the proxy loss weight λ_t are linearly annealed from 0 to their target values over a warm-up period of W epochs:

$$p_t = \min\left(1, \frac{t}{W}\right) \cdot p_{\max}, \qquad \lambda_t = \min\left(1, \frac{t}{W}\right) \cdot \lambda_{\max}, \tag{5}$$

where t denotes the current epoch. This gradual introduction ensures that the detector first learns robust representations from high-quality base probability lists before progressively adapting to proxy predictions. At inference time, all base LLMs are dropped, and the proxy-produced $\hat{\mathbf{P}}_M$ is directly used as $\mathbf{S}^*_M$.

We then use convolutional neural networks and Transformer encoders to extract features $\mathbf{R}_F \in \mathbb{R}^{M \times d}$ from $\mathbf{S}^*_M$, where d denotes the feature dimension.

Based on the feature source used during inference, we refer to two versions of PhantomHunter: **PhantomHunter$_{\text{Base}}$** directly uses the base probability lists $\mathbf{P}_M$ extracted from the M base LLMs as input features. **PhantomHunter$_{\text{Proxy}}$** replaces the base LLMs with the lightweight proxy module at inference time, using the proxy-predicted $\hat{\mathbf{P}}_M$ as input features.

3.2 Contrastive Family-Aware Learning

To better model family relationships in the feature space and enhance the detector's generalization within the same family, we treat samples from the same family as augmentations and employ contrastive learning to enhance the representation $\mathbf{R}_F$. Specifically, within each mini-batch, texts generated by models from the same family as the original text's model are treated as positive examples, while others are negative ones. We compute a SimCLR-style [14] contrastive loss $\mathcal{L}_C$:

$$\mathcal{L}_C = -\sum_{i \in b} \log \frac{\exp(\delta(\mathbf{R}^m_{F_i}, \tilde{\mathbf{R}}^m_{F_i})/t)}{\sum_{j=1}^{2|b|} \mathbf{1}_{[j \neq i]} \exp(\delta(\mathbf{R}^m_{F_i}, \mathbf{R}_{F_j})/t)}, \tag{6}$$

where $\mathbf{R}^m_{F_i}/\tilde{\mathbf{R}}^m_{F_i}$ are the embeddings of the original/augmented sample, respectively, δ is a dot-product function, and t controls the temperature. This contrastive objective optimizes the trainable CNN–Transformer feature encoder in Step 1 that maps $\mathbf{S}^*_M$ to $\mathbf{R}_F$. We use a multi-layer perceptron (MLP) to classify the LLM family:

$$\hat{y}_F = \text{softmax}(\text{MLP}_\text{F}(\mathbf{R}_F)), \tag{7}$$

where $\hat{y}_F \in \{\theta_1, \theta_2, \cdots, \theta_M\}$ represents the predicted LLM family.

3.3 Mixture-of-Experts-Based Detection

Finally, we employ a mixture-of-experts network for binary classification of human versus AI-generated text. Each expert detector specializes in detecting texts from specific LLM families, enhancing detection performance. Specifically, we use the family prediction result $\hat{y}_F$ as a gating signal to control the weighting of each expert's prediction, yielding the final binary judgment:

$$\hat{y}_B = \sum_{i=1}^{M} \hat{y}^i_F \cdot \text{softmax}(\text{MLP}_{B_i}(\mathbf{R}_F)), \tag{8}$$

Table 1 Statistics of the corpora for fine-tuning.

ArXiv/Q&A Domain	# Tokens
Computer Science (cs)	6,441,516
Physics (phy)	5,955,389
Others (oth; q-bio/stat/eess/math/q-fin/econ)	3,939,626
ELI5	202,658
Finance (fin)	234,787
Medicine (med)	254,292

where $\hat{y}_F^i$ represents the probability that the text belongs to the family of base model θ_i according to the family classifier. $\hat{y}_B \in [0, 1]$ denotes the probability of the $\mathbf{x}$ being LLM-generated. We optimize the PhantomHunter model with:

$$\mathcal{L} = \lambda_1(\mathcal{L}_F + \tilde{\mathcal{L}}_F) + \lambda_2\mathcal{L}_B + \lambda_3\mathcal{L}_C + \lambda_4\mathcal{L}_{proxy}, \quad (9)$$

where $\mathcal{L}_F$ and $\tilde{\mathcal{L}}_F$ represent the cross-entropy losses for family classification of $\mathbf{x}$ and $\tilde{\mathbf{x}}$ respectively, $\mathcal{L}_B$ is the cross-entropy loss for LLM text detection, $\mathcal{L}_C$ is the contrastive loss, $\mathcal{L}_{proxy}$ is the MSE distillation loss. λ_i $(i = 1, 2, 3, 4)$ are hyperparameters to balance the loss items during the optimization. As shown in Figure 4, the updated parameters include: (1) the light weight proxy and the CNN–Transformer feature encoder after the probability-list input in Step 1, (2) the family predictor used for family-aware learning and gating in Step 2, and (3) the family-specific detectors in the MoE module during Step 3.

PhantomHunter captures inherent signatures that persist through fine-tuning by learning family-level features from observable fine-tuned models derived from the same base LLMs, enabling the detection of unseen models from known families. Rather than focusing on individual model characteristics, we emphasize shared patterns defining model families. This family-aware approach allows the detector to generalize to unseen fine-tuned models by recognizing persistent probability patterns that remain after extensive fine-tuning. The method is effective because fine-tuning primarily adapts a model's behavior to specific domains while preserving much of the underlying probabilistic structure inherited from the base model.

4 Experiment

In this section, we conduct experiments to answer the following questions:

EQ1: How effective is PhantomHunter at detecting texts from unseen fine-tuned LLMs?

EQ2: How do different components of PhantomHunter contribute to its overall performance?

EQ3: How does PhantomHunter perform in diverse real-world settings?

4.1 Dataset Construction

We fully simulate the two most common usage scenarios of LLM: writing and question-answering. For writing, we collect 69,297 abstracts of academic papers from the arXiv archive[2], categorizing them by primary subjects (domains). For Q&A, we collect 3,062 Q&A pairs in ELI5, finance, and medicine domains from HC3 dateset [5].

[2] https://arxiv.org/archive/

Table 2 Statistics of the evaluation dataset for privately-tuned LLMGT detection. FT Domain: Domains of data for LLM fine-tuning ("base" denotes no fine-tuning).

	Dataset	Source	FT Domain	#Samples
arXiv	Train	LLaMA	base/phy/oth	1,564/1,969/1,971
		Gemma	base/phy/oth	1,604/1,455/1,457
		Mistral	base/phy/oth	1,778/1,877/1,826
		Qwen	base/phy/oth	1,640/1,546/1,543
		Human	-	1,145
	Test	LLaMA	cs	2,076
		Gemma	cs	1,540
		Mistral	cs	2,002
		Qwen	cs	2560
		Human	-	208
Q&A	Train	LLaMA	base/med/ELI5	680/256/220
		Gemma	base/med/ELI5	680/256/220
		Mistral	base/med/ELI5	680/256/220
		Qwen	base/med/ELI5	680/256/220
		Human	-	5,780
	Test	LLaMA	fin	204
		Gemma	fin	204
		Mistral	fin	204
		Qwen	fin	204
		Human	-	806
		Total		32,818

Table 1 shows the corpora size. Note that we merge subjects other than physics and computer science for the arXiv part to balance fine-tuning scales across domains. We select open-source LLaMA-2 7B-Chat [15], Gemma 7B-it [16], Mistral 7B-Instruct-v0.1 [17],and Qwen2.5-7B-Instruct [18] as base models. For each scenario, we fine-tuned each base model using full-parameter fine-tuning (Full) and LoRA fine-tuning on the corresponding 3 domain-specific corpora from arxiv and Q&A, resulting in 48 derivative models. The details are as follows:

- **Full fine-tuning** was performed at the `sft` (supervised finetuning) stage. The training configuration was set with a batch size of 1 per device and gradient accumulation steps of 4. The learning rate was set to 1.0e-5, and the maximum number of training epochs was 10. A cosine learning rate scheduler was used with a warm-up ratio of 0.1. BF16 precision was enabled for efficient training, and the DDP (Distributed Data Parallel) strategy was employed.
- **LoRA fine-tuning** [7] was performed with the LoRA rank set to 8, and the training was conducted with a batch size of 1 per device, gradient accumulation steps of 4, a learning rate of 1.0e-5, and maximum number of epochs of 10, utilizing a cosine learning rate scheduler with a warm-up ratio of 0.1, BF16 precision for efficient training.

We then generate abstracts with 4 base models and 12 arXiv-based derivative models, and answers with the same base models and 12 Q&A-based derivative ones. For the arXiv dataset, we designate LLMs fine-tuned on computer science (cs) data as unseen ones, reserving them exclusively for testing. And for the Q&A dataset, we designate LLMs fine-tuned on finance (fin) data as unseen ones, which means it

Table 3 F1 scores for human-written (Human), LLM-generated (Gen), and both (Macro) and AUC for unseen fine-tuned LLM-generated text detection. The two best results in each metric are **bolded** and underlined. $\text{TDT}_{\text{Threshold}}$ applies a threshold to the TDT score, while TDT_{SVR} trains an SVR using three wavelet-band features. $\text{PhantomHunter}_{\text{Base}}$ uses probability lists from four base LLMs, while $\text{PhantomHunter}_{\text{Proxy}}$ uses the default MSE-trained lightweight proxy.

Method	arXiv								Q&A							
	Full				LoRA				Full				LoRA			
	Human	Gen.	Macro	AUC	Human	Gen.	Macro	AUC	Human	Gen.	Macro	AUC	Human	Gen.	Macro	AUC
Training-Based Methods																
RoBERTa	62.42	98.54	80.48	97.63	65.61	98.66	82.13	93.16	84.30	95.12	89.71	98.72	58.12	78.03	68.07	99.75
T5-Sentinel	12.81	68.12	40.47	1.30	68.99	46.67	57.83	1.08	84.92	81.68	83.30	3.57	84.67	79.89	82.28	7.42
SeqXGPT	81.21	99.52	90.37	91.27	93.15	99.77	96.46	70.37	85.38	97.27	91.33	94.18	48.28	79.13	63.71	95.18
DALD	14.13	95.84	54.98	72.81	17.76	96.86	57.31	78.97	31.58	65.49	48.53	51.18	37.89	75.92	56.91	60.74
DeTeCtive	85.09	96.56	90.83	97.27	87.66	99.64	93.65	91.42	89.08	96.84	92.96	96.54	71.58	88.98	80.28	99.21
TDT_{SVR}	53.52	98.79	76.15	90.76	88.79	99.71	94.25	99.00	75.98	93.05	84.51	93.03	66.67	89.58	78.12	90.92
Training-Free Methods																
DNA-GPT	66.29	95.37	80.83	76.64	72.08	99.02	85.55	69.09	68.10	73.14	70.62	86.04	64.84	66.02	65.43	96.33
DetectGPT	26.71	95.50	61.11	85.56	52.53	98.33	75.43	79.27	59.46	88.72	74.09	82.80	26.09	95.58	61.11	92.36
Fast-DetectGPT	36.82	96.18	66.50	96.04	90.82	99.78	95.30	93.27	81.03	93.90	87.47	96.62	75.32	90.10	82.71	99.92
$\text{TDT}_{\text{Threshold}}$	0	98.73	49.36	49.94	91.78	99.79	95.79	99.45	82.51	95.11	88.81	96.11	78.79	94.89	86.84	95.07
Our Method: PhantomHunter																
$\text{PhantomHunter}_{\text{Base}}$	85.59	99.60	92.59	99.12	**95.81**	**99.89**	**97.85**	98.22	**90.67**	**97.36**	**94.01**	**99.70**	**89.91**	**97.26**	**93.58**	**99.99**
$\text{PhantomHunter}_{\text{Proxy}}$	**91.74**	**99.77**	**95.76**	**99.91**	91.15	99.75	95.45	**99.98**	90.10	97.24	93.67	97.85	87.89	96.61	92.25	98.66

is only used as a test dataset. Detailed statistics of our experimental dataset are shown in Table 2, and the prompt templates are as follows:

> **Prompt Description**: [Generating an abstract based on the title of an academic paper, used in the arxiv scenario]
> **Instruction Prompt**: [Write an abstract for the academic paper titled [*title*].]

> **Prompt Description**: [Generating an answer based on the given question, used in the Q&A scenario]
> **Instruction Prompt**: [*question*]

4.2 Experimental Setup

Implementation Details. In PhantomHunter, the CNN encoder comprises three convolutional layers followed by max-pooling, and the Transformer encoder has two attention layers with four attention heads each. The feature dimension d is set to 128. In detail, the probability features are first fed into a five-layer one-dimensional convolutional network $f : L \to Z$. This module extracts local latent representations through specific kernel sizes (5, 3, 3, 3, 3) and uniform strides (1, 1, 1, 1, 1), with channel dimensions varying across layers (64, 128, 128, 128, 64). Subsequently, the output is passed to a contextual network $g : Z \to C$, which consists of two Transformer encoder layers configured with 4 attention heads and a 256-dimensional feed-forward network. Equipped with fixed positional embeddings, this module leverages self-attention mechanisms to capture contextual features. For contrastive learning, we use a temperature t of 0.07. The hyperparameters in the loss function are set as $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, and $\lambda_3 = 0.5$, respectively. The model is trained using the Adam optimizer with a learning rate of 2e-5 and a batch size of 32. We train for 10 epochs and select the best-performing checkpoint based on the validation set.

Evaluation Metrics. We report F1 scores on each testing subset (i.e., human-written and LLM-generated subsets) with a threshold of 0.5 and the macro F1. We also report the threshold-free metric AUC. For comparison with commercial detectors, we add the true positive rate under a 1% false positive rate to align with real-world scenarios, following the recommendation from Tufts et al. [19].

Compared Baselines. We select the eight LLM-generated text detection methods for comparison, covering both training-based and training-free methods: **RoBERTa** [5], **T5-Sentinel** [20], **SeqXGPT** [10], **DNA-GPT** [21], **DetectGPT** [22], **Fast-DetectGPT** [23], **TDT** [24], **DALD** [25], **DeTeCtive** [26]. The settings for all baselines are detailed in the appendix.

4.3 Main Results (EQ1)

Table 3 presents the binary classification results (human vs. LLM-generated) of PhantomHunter and baselines on unseen fine-tuned models. We have the following observations:

1) Compared to all baselines, PhantomHunter demonstrates superior performance in detecting texts from unseen fine-tuned models. Across all four settings, both the base and proxy versions of PhantomHunter consistently achieve Macro F1 scores above 92%, showing balanced detection capability for both human-written and LLM-generated texts. For full fine-tuning, PhantomHunter improves the Macro F1 score over the best baseline by 4.93% and 1.05% on arXiv

Table 4 Ablation results averaged over arXiv-Full, arXiv-LoRA, Q&A-Full, and Q&A-LoRA. "w/ KL" replaces MSE with KL divergence for proxy distillation. BFE: Base probability feature extraction. CL: Contrastive learning. MoE: Mixture-of-experts. The best results in each metric are **bolded**.

Method	Human F1	Gen. F1	Macro F1	AUC
Base Version				
PhantomHunter$_{\text{Base}}$	**90.50**	**98.53**	**94.51**	99.26
w/o BFE	71.36	95.68	83.52	**99.73**
w/o CL	83.88	97.13	90.51	98.27
w/o MoE	81.92	96.79	89.36	98.11
Proxy Version				
PhantomHunter$_{\text{Proxy}}$	**90.22**	**98.34**	**94.28**	99.10
w/ KL	86.87	98.34	92.60	98.91
w/o BFE	69.79	93.18	81.49	99.01
w/o CL	79.58	94.11	86.84	98.78
w/o MoE	87.77	97.66	92.71	**99.16**

Table 5 Performance of commercial detectors (%). The two best results are **bolded** and underlined, respectively. Here, PhantomHunter refers to the light-weight proxy version.

Method/Service	TPR$_{1\%\text{FPR}}$	F1$_{\text{Human}}$	F1$_{\text{Gen}}$	F1$_{\text{Macro}}$
AIorNot	8.02	23.73	57.91	40.82
Aliyun	9.98	23.46	52.52	37.99
BlueEyes	34.23	50.61	89.54	70.08
Copyleaks	25.51	23.12	51.43	37.28
GPTZero	25.52	23.02	51.55	37.28
Originality.ai	25.51	23.01	51.50	37.25
Phrasly.ai	29.89	21.45	41.69	31.57
Sapling	24.66	60.80	94.76	77.78
TencentCloud	<u>69.32</u>	<u>73.57</u>	<u>96.71</u>	<u>85.14</u>
Turnitin	25.59	22.98	51.59	37.29
ZeroGPT	25.51	23.34	51.58	37.46
PhantomHunter	**91.18**	**90.22**	**98.34**	**94.28**

and Q&A, respectively; for LoRA fine-tuning, the improvements are 1.39% and 6.74%, respectively. This exhibits PhantomHunter's detection capability for texts generated by unseen fine-tuned LLMs.

2) The fine-tuning approaches do not present consistent detectability for different domains or detectors. In most cases, the generated arXiv texts from LoRA-tuned LLMs are easier to detect than those from full-tuned ones, while the situations are reversed for Q&A texts. This difference may be attributed to the greater length of arXiv texts, which could better reflect the effects of full parameter changes brought by full fine-tuning. In such a comparison, PhantomHunter still keeps balanced performance.

3) Some detectors show relatively low performance for the human class, regardless of the class-wise balance of the training sets, indicating their difficulties in identifying human-written texts. We attribute this failure to their inherent ignorance of family traits shared among the base and derivative LLMs, which may shape a relatively blurred boundary between human and LLM texts.

4) PhantomHunter$_{\text{Proxy}}$ achieves competitive performance while reducing inference costs. Compared with PhantomHunter$_{\text{Base}}$, the proxy version avoids forwarding each input through the four base LLMs and instead uses a lightweight proxy to approximate the probability lists. Despite this cost reduction, its performance remains close to the base version. This indicates that the proxy can preserve most family-aware detection signals while making PhantomHunter more practical for deployment.

4.4 Ablation Studies (EQ2)

To evaluate the contribution of each component, we conduct ablation studies on both the base and proxy versions of PhantomHunter. All ablation variants follow the same training and evaluation protocol as the main experiments, and we report the average results over four evaluation settings: arXiv-Full, arXiv-LoRA, Q&A-Full, and Q&A-LoRA. For the base version, we remove the base probability feature extractor (*w/o* BFE), contrastive learning (*w/o* CL), and the mixture-of-experts module (*w/o* MoE), respectively. For the proxy version, we apply the same component-level ablations and further evaluate the effect of the proxy distillation objective by replacing the default MSE loss with KL divergence (*w/* KL).

For the KL-loss variant, we follow the standard temperature-scaled knowledge distillation formulation [27, 28] with $\tau = 2.0$. Specifically, we define the temperature-scaled base and proxy distributions as

$$q^{(\tau)} = \sigma\left(\frac{\mathbf{P}_M}{\tau}\right), \qquad p^{(\tau)} = \sigma\left(\frac{\hat{\mathbf{P}}_M}{\tau}\right), \tag{10}$$

where $\sigma(\cdot)$ denotes the softmax operation along the token dimension, and $\mathbf{P}_M$ and $\hat{\mathbf{P}}_M$ follow the same definitions as in Section 3. The KL distillation loss is computed as:

$$\mathcal{L}_{\text{KL}} = \tau^2 \cdot \frac{1}{B}\sum_{b=1}^{B}\sum_{m=1}^{M}\sum_{n=1}^{N} q_{bmn}^{(\tau)} \log \frac{q_{bmn}^{(\tau)}}{p_{bmn}^{(\tau)}}. \tag{11}$$

Here, B, M, and N denote the batch size, the number of base LLMs, and the truncated sequence length, respectively, while b, m, and n are their corresponding indices. The factor τ^2 follows standard temperature-scaled distillation and is used to preserve the gradient scale. The results are shown in Table 4 and the corresponding PR curves are visualized in Figure 5.

From the results, we observe that removing any component leads to a degradation of detection performance, especially for BFE, since it is the probabilistic characterization of the base model that embodies the family commonality. Without contrastive loss, the probability representations of different family-generated texts become less distinguishable in the feature space. Due to the removal of the mixture-of-experts module, the classifier is forced to find a decision boundary between *all* family-generated texts and human texts without specializing in particular family models, which makes the learning process more difficult and thus lowers the detection performance. Replacing the default MSE distillation loss with KL divergence reduces Macro F1 from 94.28 to 92.60, suggesting that directly matching token-level probability lists with MSE provides a more stable proxy learning signal in our setting.

4.5 Further Analysis (EQ3)

Comparison with Commercial Detectors. To validate the real-world difficulties of text from privately-tuned LLMs, we test 11 accessible

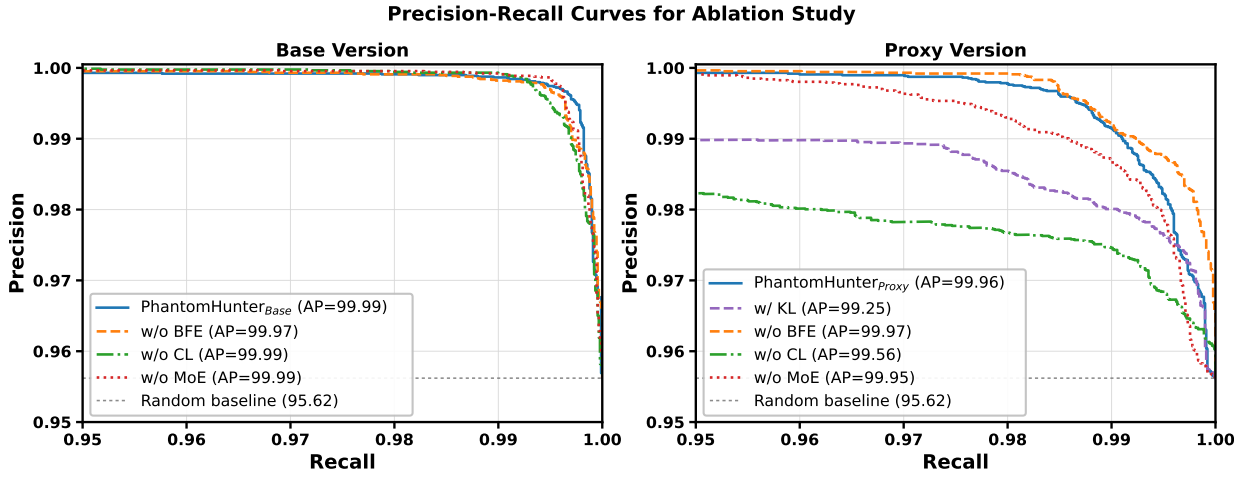

Fig. 5 Precision-recall curves for the ablation study. The curves are computed by pooling the predictions over arXiv-Full, arXiv-LoRA, Q&A-Full, and Q&A-LoRA, and are shown in the high-precision and high-recall region to better visualize the differences among ablation variants.

commercial detectors that support API calling with the whole test set (Appendix 6). Due to the inaccessibility, we do not consider the commercial detectors with the requirement of additional actions besides API callings (*e.g.*, Grammarly requires a subscription to the enterprise plan first) or have strict constraints in request times (*e.g.*, Zhuque AI Detector allows only 5 requests per day). We remove the invalid samples for each detector.

We assume that commercial detectors have learned the traits of the base models of involved families and been trained on a larger and more diverse training corpus than ours, but were not specially trained for defending against fine-tuning attacks. However, Table 5 shows that PhantomHunter, trained on a small but focused dataset instead, significantly outperforms the compared ones, again highlighting the value of the proposed family-aware learning. We acknowledge that this is not a completely fair comparison, but it shows the real-world impacts of privately-tuned LLMs and how PhantomHunter may survive under such attacks.

Generalizability Test on the Public Benchmark DetectRL To evaluate the performance of our framework on the public benchmark, we conduct *direct* inference of PhantomHunter and compare baselines trained on our corpus only against the representative public benchmark DetectRL [29], which covers diverse LLMs and multiple domains. The setting encompasses 4 distinct domains, including academic abstracts from arXiv (specifically, completely different from the arXiv data in the training set), news document summaries from XSum, creative writing stories from Writing Prompts, and restaurant reviews from Yelp. Each domain contains 1,008 samples, resulting in a total of 4,032 test instances. This balanced multi-domain setting enables a comprehensive evaluation of detection performance across different content types and writing styles. We followed the parameter configurations and feature extraction process from the main experiments described in our paper. We did not perform any training on this dataset and only conducted inference. From the results in Table 6, we see that PhantomHunter achieves a macro F1 score of 80.22% (84.82% for human-written text and 76.15% for LLM-generated text). Across the multi-domain split (arXiv, WritingPrompts, XSum, and Yelp), it retains an average macro F1 score of 77.1%, confirming reliable cross-domain generalization without domain-specific tuning.

A Practice of Including LLMs beyond Known Families. In this section, we explore how to extend our framework to detect other families (e.g., ChatGPT, Claude, and other closed-source LLMs) where base model features are not available. In practice, AIGT detectors should maintain good performance against both known and unknown family models. Here, we show that PhantomHunter can be easily extended to be compatible with LLMs beyond known families through a simple modification. Specifically, an additional category "others" is added to the label space of the family classifier MLP_F, which is used as the family label of these texts from models beyond known families, and the number of experts in MoE is increased accordingly. In detail, we conduct two experiments. 1. We introduced GPT-4o mini[3] and Claude-3.7 Sonnet[4] to generate texts, which were labeled as "others". These were used for training and testing. 2. Based on the experiment above, DeepSeek-v3[5] and GLM-4-Air[6] were further introduced during the evaluation phase as entirely new LLM families that were unseen in training. Moreover, to further evaluate generalization to different versions within the same series, we additionally introduced Qwen3-8B-Base[7] during the evaluation phase as an entirely new LLM that was unseen in training. In addition, the settings followed those of the main experiment. Table 7 shows that PhantomHunter maintains high performance not only for private fine-tuning LLMs but also for GPT-4o mini and Claude 3.7 Sonnet, indicating its good compatibility and extendibility to more diverse LLMs. For new, entirely unseen LLMs (*i.e.,* DeepSeek-v3, GLM-4-Air, and Qwen3-8B-Base), Phan-

[3] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/
[4] https://www.anthropic.com/news/claude-3-7-sonnet
[5] https://github.com/deepseek-ai/DeepSeek-V3
[6] https://github.com/zai-org/GLM-4
[7] https://huggingface.co/Qwen/Qwen3-8B

Table 6 Generalization performance comparison among different methods (%), tested on the DetectRL benchmark. The two best results are **bolded** and underlined, respectively.

Method	$F1_{Human}$	$F1_{Gen}$	$F1_{Macro}$
RoBERTa	00.00	78.71	39.36
T5-Sentinel	26.87	57.32	42.09
SeqXGPT	81.04	67.25	74.15
DetectGPT	61.90	54.91	58.41
DNA-GPT	67.58	60.77	64.18
FastDetectGPT	74.04	64.65	69.35
DeTeCtive	48.17	63.86	56.02
PhantomHunter	**84.82**	**76.15**	**80.22**
w/o BFE	56.21	71.20	63.70
w/o CL	75.45	67.29	71.37
w/o MoE	64.64	72.63	68.64

tomHunter also demonstrates strong generalization capabilities.

Table 7 Performance on LLMs beyond known families by adding the "others" family class. PhantomHunter generalizes to GPT-4o mini, Claude-3.7 Sonnet, entirely unseen LLM families (DeepSeek-v3 and GLM-4-Air), and different versions within the same series Qwen3-8B-Base.

Model	$TPR_{1\%FPR}$	$F1_{Human}$	$F1_{Gen}$	$F1_{Macro}$
Known families (seen during training)				
LLaMA-cs	99.90	97.56	99.76	98.66
Mistral-cs	99.90	98.04	99.80	98.92
Gemma-cs	99.87	97.56	99.68	98.62
Qwen-cs	98.98	93.02	99.41	96.22
Others (seen during training)				
GPT-4o mini	99.67	97.56	99.17	98.37
Claude 3.7 Sonnet	100.00	98.04	99.50	98.77
Others (unseen during training)				
DeepSeek-v3	99.40	98.04	98.11	98.08
GLM-4-Air	99.77	98.04	98.11	98.07
Qwen3-8B-Base	97.12	96.52	96.74	96.63

Robustness to Larger Fine-tuning Scales. To further examine whether family-level traits remain useful under stronger fine-tuning, we evaluate PhantomHunter on checkpoints fine-tuned with larger corpora, ranging from 25M to 100M tokens. As shown in Table 8, PhantomHunter remains stable under LoRA fine-tuning and full fine-tuning on arXiv-CS, while full fine-tuning on Q&A-finance is more challenging and represents a boundary case. Even under the 100M-token setting, the results indicate that family-aware signals are not fully erased by larger-scale fine-tuning in our evaluated settings.

Deployment Efficiency and Cost Analysis. In practical black-box detection applications, deployment cost is a key bottleneck. We therefore compare the training and inference costs of PhantomHunter with representative baselines, focusing on the trade-off between one-time training overhead and repeated inference cost in real-world deployment.

1) Experiment Setup: For the efficiency evaluation, we measure

Table 8 Robustness of PhantomHunter under larger fine-tuning scales

Type	Dataset	25M	50M	75M	100M
LoRA	arXiv-CS	94.30	93.98	92.58	91.09
	Q&A-finance	94.55	94.55	99.09	96.36
Full	arXiv-CS	97.42	96.72	97.81	97.50
	Q&A-finance	90.00	90.00	85.45	85.45

Table 9 Inference efficiency comparison with training-based baselines.

	Method	**Latency** (ms)	**GPU Memory** (GB)
Baseline	RoBERTa	9.45	0.51
	DeTeCtive	9.84	1.00
	DALD	92.02	26.89
	T5-Sentinel	118.13	11.66
	SeqXGPT	20,829.45	59.16
Ours	$PhantomHunter_{Base}$	20,799.25	59.77
	$PhantomHunter_{Proxy}$	14.29	0.63

the end-to-end inference latency and peak GPU memory usage of each method using 256-token inputs on a single NVIDIA A800 GPU. For RoBERTa and DeTeCtive, we report the cost of their encoder-based classification pipelines, where RoBERTa uses a RoBERTa-base binary classifier, and DeTeCtive uses its SimCSE-RoBERTa-based feature extractor with the downstream prediction module. For T5-Sentinel, we measure the full T5-based detector inference pipeline. For DALD, we follow its original surrogate-model setting and measure inference with the aligned LLaMA2-7B surrogate model. For SeqXGPT, we report the full end-to-end cost, including autoregressive probability-feature extraction with the required surrogate LLMs and the subsequent classifier. For $PhantomHunter_{Base}$, the latency and memory include probability feature extraction from the four deployed base LLMs and the family-aware detector. For $PhantomHunter_{Proxy}$, the four base LLMs are replaced by the lightweight proxy module, so inference only requires the RoBERTa-base encoder, the two-layer MLP, and the family-aware detector. All methods use the same parameter settings as in the main experiments.

2) Training Cost Trade-offs: PhantomHunter incurs additional computation during training because the proxy needs supervision from the base LLMs. Specifically, fitting the proxy targets requires deploying four 7B-parameter LLMs to perform sampling and obtain real logits. In addition, the proxy module, consisting of an approximately 125M-parameter RoBERTa encoder and a two-layer MLP, is trained to learn and project the proxy representations. With all base models loaded in FP16 precision, a training batch size of 64, and a maximum proxy sequence length of 256 tokens, the training process requires approximately 60 GB of GPU memory. Notably, this cost is paid during training and proxy construction, rather than during deployment.

3) Inference Cost Advantages: The main advantage of the proxy module appears at inference time. Many probability-based or feature-based detectors require expensive model queries for each input. For example, DALD relies on a 7B-parameter surrogate LLM, T5-Sentinel requires a large T5-style detector, and SeqXGPT requires deploying four proxy LLMs to extract autoregressive sequence features. Similarly,

Table 10 F1 scores for family classification of unseen fine-tuned LLM-generated texts.

Dataset		LLaMA	Mistral	Gemma	Qwen	$F1_{Macro}$
arXiv	Full	75.61	63.71	95.76	87.21	80.57
	LoRA	78.32	77.30	56.63	79.11	72.84
Q&A	Full	82.63	50.29	66.29	56.89	64.03
	LoRA	68.21	50.18	71.82	61.10	62.83

Table 11 Linguistic statistics for human reference texts and misclassified model-generated samples on the *arXiv-Full* test subset. The results are computed over the 1214 samples misclassified by the family classifier. TTR denotes the percentage of unique words in the total number of words, AvgLen denotes the average number of characters per word, and Punct, Clause, Modal, and Pron denote punctuation marks, approximate clause markers, modal verbs, and pronouns per 100 words, respectively.

Family	TTR	AvgLen	Punct	Clause	Modal	Pron
Human	0.711	5.98	13.06	7.56	0.40	4.67
Gemma	0.954	5.86	10.91	5.99	0.85	5.14
Qwen	0.586	5.60	11.06	8.15	0.76	6.52
LLaMA	0.751	5.76	12.04	7.66	0.60	6.59
Mistral	0.735	5.71	12.20	7.79	0.62	6.63

$\text{PhantomHunter}_{\text{Base}}$ needs forward passes through four base LLMs, resulting in a large inference footprint. In contrast, $\text{PhantomHunter}_{\text{Proxy}}$ removes this dependency by replacing the four base LLMs with a lightweight RoBERTa-base encoder and a two-layer MLP. As shown in Table 9, this reduces latency from 20,799.25 ms for $\text{PhantomHunter}_{\text{Base}}$ to 14.29 ms for $\text{PhantomHunter}_{\text{Proxy}}$ on 256-token inputs, while GPU memory decreases from 59.77 GB to 0.63 GB. Compared with SeqXGPT, which has a similar latency and memory footprint to the base version of PhantomHunter, the proxy version reduces latency by more than three orders of magnitude and requires less than 1 GB GPU memory. Although RoBERTa and DeTeCtive are slightly faster, $\text{PhantomHunter}_{\text{Proxy}}$ remains in the same low-latency range while providing stronger robustness to unseen fine-tuned open-source LLMs, shown in Table 3. These results indicate that the proxy design effectively amortizes the training-time cost and makes PhantomHunter practical for repeated large-scale inference.

4.6 Exploration of Source Family Prediction

Results of Source Family Prediction. Besides enhancing the learning of the LLMGT detection head, the introduction of a family classifier makes it possible to provide a source family prediction simultaneously, which facilitates further forensics analysis and may improve users' trustworthiness in the detector. Table 10 previews the current performance of known family prediction. The performance is mixed. The highest F1 score exceeds 95% while the lowest is below 55%, demonstrating that the family prediction is far more challenging than LLMGT detection and the current design requires improvement for family prediction. Recalling the main results, this also implies that even a moderate family-aware learning could bring performance improvement, indicating that its potential remains to be further explored.

Case Analysis. To better understand the limitations of the current family classifier, we examine representative misclassified samples from the *arXiv-Full* subset and contextualize them with linguistic statistics of the misclassified cases. As shown in Table 11, the Human row is computed over all human reference texts, while each model-family row is computed only over samples misclassified by the family classifier. All statistics are first computed at the sample level and then averaged within each group. Word tokens are extracted using a regular-expression-based tokenizer over alphabetic, hyphenated, and apostrophe-containing forms. Our observations reveal several characteristic patterns:

1) The Gemma family exhibits highly distinctive and simplified syntactic behavior. Compared with human, LLaMA, Mistral, and Qwen texts, Gemma shows the highest lexical diversity (TTR = 0.954) and lower punctuation and clause-marker densities (Punct = 10.91, Clause = 5.99). This extreme combination of "high diversity + low structural complexity" results in a unique feature island that the detector can recognize reliably.

2) The Qwen family shows a different signature. It has the lowest lexical diversity among all families (TTR = 0.586), indicating stronger lexical reuse, and it also uses shorter words on average (AvgLen = 5.60). In addition, Qwen exhibits relatively high clause-marker density (Clause = 8.15) and modal density (Modal = 0.76), producing a characteristic lexical-syntactic profile that helps the classifier isolate this family.

3) For LLaMA and Mistral families, their clause-marker density and punctuation usage closely resemble human texts. This presents the most challenging cases. To further visualize this overlap, Table 12 presents three same-prompt examples, including a human text and two model-generated samples from LLaMA and Mistral. The highlighted cues correspond to the linguistic dimensions reported in Table 11, including token reuse, clause/discourse markers, punctuation, and pronouns. Their aggregate statistics are close to human writing across multiple dimensions: clause density (7.66 and 7.79 vs. 7.56 for human), punctuation density (12.04 and 12.20 vs. 13.06), lexical richness (TTR 0.751 and 0.735 vs. 0.711), and average word length (5.76 and 5.71 vs. 5.98). Because no single surface feature distinctly differentiates these models from humans, the classifier often confuses them with each other.

Overall, Gemma and Qwen present more distinctive aggregate feature profiles, whereas LLaMA and Mistral cluster more closely with human writing across multiple linguistic dimensions. This helps explain why the classifier performs more reliably on the former and produces more confusion on the latter.

■ 5 Related Works

LLM-Generated Text Detection.

LLM-generated text detection has become an important research topic as LLMs are increasingly used in writing, education, news production, DeepResearch. From a corpus-linguistic perspective, LLMs deviate from human text in systematic lexico-grammatical patterns, and the degree of deviation varies across models, domains, and registers [30, 31]. This observation motivates a broad line of studies that attempt to distinguish human-written and machine-generated text.

Existing methods can be categorized as model-based and metric-based ones. Model-based approaches train neural detectors to distin-

Table 12 Three examples from the *arXiv-Full* subset. Blue marks TTR reused tokens, light blue marks approximate clause/discourse cues, pink marks punctuation cues, and red marks pronouns.

#	**Prompt: Write an abstract for the academic paper titled 'FAR: Flexible, Accurate and Robust 6DoF Relative Camera Pose Estimation.**	
1	Estimating relative camera poses between images has been a central problem in computer vision . Methods that find correspondences and solve for the fundamental matrix offer high precision in most cases . Conversely , methods predicting pose directly using neural networks are more robust to limited overlap and can infer absolute translation scale , but at the expense of reduced precision . We show how to combine the best of both methods ; our approach yields results that are both precise and robust , while also accurately inferring translation scales . At the heart of our model lies a Transformer that (1) learns to balance between solved and learned pose estimations , and (2) provides a prior to guide a solver . A comprehensive analysis supports our design choices and demonstrates that our method adapts flexibly to various feature extractors and correspondence estimators , showing state-of-the-art performance in 6DoF pose estimation on Matterport3D , InteriorNet , StreetLearn , and Map-free Relocalization .	
	Human	**Words:** 147 **TTR:** 0.73
2	Estimating relative camera poses between images has been a central problem in computer vision . Methods using deep learning have achieved state-of-the-art performance , but are generally restricted to relatively high-resolution images . We present DeepRP , an approach for estimating relative pose from RGB and depth images , which overcomes the limited resolution of prior methods by exploiting multi-scale information , enabling it to run on much lower-resolution inputs . The method uses two separate branches : one with a ResNet34 backbone trained solely on depth data and another with a ResNet18 backbone that also utilizes RGB information . In addition , we introduce a novel self-supervised training technique called Sparse Supervision (Sup) , which efficiently leverages both types of information while	
	LLaMA	**Words:** 113 **TTR:** 0.80
3	Estimating relative camera poses between images has been a central problem in computer vision . Methods such as SfM and multi-view stereo (MVS) have long relied on the assumption that images are rectified , i.e. , each camera's principal point is at the center of its image . This assumption is common in many applications and makes computation easier by ensuring that the epipolar geometry of the images is invariant to scale . However , the rectification assumption does not always hold in practice , especially for uncalibrated cameras or when using off-the-shelf hardware . In this paper , we investigate how to handle non-rectified images in existing pipelines . We introduce a novel perspective-n-point (PnP) algorithm that solves for both the absolute orientation and position of an image while allowing	
	Mistral	**Words:** 124 **TTR:** 0.77

guish human from LLM-generated text by analyzing textual characteristics. Early work fine-tunes pretrained models [32, 33] on labeled corpora, capturing textual characteristics. More recent methods pursue richer representations, exploiting semantic features [5, 26, 34] and self-familiarity [35]. Li et al. [36] further study detection in the streaming content monitoring, moving from post-hoc judgment to early intervention by identifying harmful LLM outputs during generation and stopping unsafe continuations in real time. Model-based methods often sets similarity metrics based on LLMs' output probability or behavioral characteristics [22, 37–39]. There are also methods that leverage LLMs themselves to derive detection signals for LLM-generated text [40, 41]. Recent methods collectively address different aspects of the LLM detection challenge, from data efficiency [42], to generalizability across models and domains [43], and training paradigms [44]. However, existing methods and evaluation benchmarks [29, 45, 46] focus on publicly known LLMs. Such ignorance leads to detectors' unsatisfying performance when countering texts from privately-tuned LLMs, as we experimentally showed before.

One possible solution is to add model-specific watermarks into LLM outputs as identifiers [47–49], which in principle allows tracing text back to its source. However, this approach imposes non-trivial infrastructure costs on LLM providers, requires coordinated adoption across the ecosystem, and remains vulnerable to adversarial attacks that can distort or remove watermark signals without substantially degrading text quality [50–52]. PhantomHunter pursues a complementary, watermark-free direction: rather than relying on embedded identifiers, it learns to detect privately-tuned LLM text by exploiting family-level distributional traits that persist across fine-tuning, and it can be combined with watermarking defenses to provide a more comprehensive detection layer.

Family Analysis of LLMs. Analyzing commonalities and uniqueness of LLMs from a family perspective yields a deep understanding of LLMs' relationship [53], which is useful for LLM attributions [6, 54] and LLMs' license violation detection [55]. As shown in [56, 57], the family factor could influence text detectability and detectors' generalizability. However, they only investigated the base models of different sizes, ignoring the unseen privately-tuned LLMs that we focus on in this research. In this work, we construct a new dataset containing texts generated by 48 LLMs that were privately fine-tuned with a domain-specific corpus with two types of fine-tuning approaches. Our study is expected to provide new knowledge about the LLMGT's detectability from the perspective of LLM post-training processes.

6 Conclusion and Discussions

We proposed PhantomHunter, which is specialized to improve the detection performance of text generated by unseen, privately-tuned LLMs. We first experimentally revealed that fine-tuning open-source base models would significantly lower the detectability of generated text, and then addressed this issue via family-aware learning to capture shared traits in each LLM family. Comparison experiments with 9 baselines and 11 industrial services confirmed PhantomHunter's superiority. Further analysis exhibited that PhantomHunter has good compatibility and extendibility to better adapt to real-world situations.

Discussions on Limitations. Though PhantomHunter showcases how to tackle the performance degradation issue when encountering texts generated by privately-tuned LLMs based on widely-known open-source ones, we also identify the following limitations of PhantomHunter, which may influence the real-world implementation. **First**, the current version only supports binary classification between human-written and LLM-generated text, plus the possible family source for a suspicious LLM-generated text piece. It does not provide fine-grained sentence- or token-level annotations [10], and the current performance of identifying family sources is moderate. **Second**, due to the experimental cost constraint, our experiment only covered several common LLM families, and the performance on other families like DeepSeek [58,59] remains unknown. **Third**, training the proxy module still requires locally deployed base LLMs to extract target features, increasing the training memory cost and computational overhead, although the deployed detector no longer requires these base LLMs at inference time.

Therefore, these directions require further exploration, and we advocate for more industrial solutions to improve the trustworthiness and efficiency of LLMGT systems. Meanwhile, it is not recommended that the feedback of PhantomHunter serve as a sole basis for real-world actions (e.g, disciplinary action against any student). Additional harmful content detection [36, 60, 61] and human verification [62] are needed.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (62406310), the Innovation Funding of ICT, CAS (E561160 and E561090), the China Postdoctoral Science Foundation (2024M763336), and the Postdoctoral Fellowship Program of CPSF (GZC20232738).

Competing Interest

The authors declare that they have no competing interests or financial conflicts to disclose.

Appendices

Introduction and Implementation of Compared Baselines

- *RoBERTa* [5]: A fine-tuned RoBERTa [32] that uses semantic features to distinguish between human and AI-generated text, which has been adopted as a typical baseline for this task.
- *T5-Sentinel* [20]: A model based on T5 [33] that treats token prediction as implicit classification to detect generated text.
- *SeqXGPT* [10]: A white-box method that treats token probability lists as waveforms and applies CNN and attention networks for detection. In the experiments, we use Mistral-7b [17], Llama-2-7b-chat [15], Qwen2.5-7B-Instruct [18], and gemma-7b [16] as white-box models to compute the probability lists.
- *DNA-GPT* [21]: A source-attribution method that leverages multiple regenerations to identify model-specific statistical fingerprints, functioning in both black-box and white-box settings. In the experiment, we adopted the white-box setting and selected Llama-2-7b-chat [15] as the proxy base model, with re-generations num $N = 10$, the truncation ratio $\gamma = 0.5$, and max new tokens num $t = 250$. Since this method is training-free, we conducted a grid search for the classification threshold with a step size of 1e-4, and reported the best result as the upper performance bound of this method.
- *DetectGPT* [22]: A zero-shot detection approach that identifies AI-generated text by comparing the probability of original passages against multiple perturbed variants without requiring model fine-tuning. In the experiment, we selected T5-3B [33] as the mask-filling model, and Llama-2-7b-chat [15] as the proxy base model, with the number of perturbations set to 100. In the experiment, we follow a similar implementation to DNA-GPT.
- *Fast-DetectGPT* [23]: An efficient improvement over DetectGPT that introduces the concept of conditional probability curvature to identify machine-generated text, while significantly reducing computational overhead through optimized perturbation sampling strategies that maintain or enhance detection performance. In the experiment, we selected gpt-neo-2.7b [63] as the sampling model and the scoring model.
- *TDT* [24]: A detection method based on *Temporal Discrepancy Tomography*, which computes token-level statistical discrepancies between a reference (base) model and a scoring model to capture how much each token deviates from the expected likelihood under the reference distribution. Following the original paper's setup, we use Llama-2-7B as the reference model and Llama-2-7B-Chat as the scoring model, with a maximum token length of 512. We evaluate two variants: $\text{TDT}_{\text{Threshold}}$, a training-free threshold-based detector using the scalar TDT score, and TDT_{SVR}, a supervised variant that trains an SVR on the development set using the three TDT wavelet-band features.
- *DALD* [25]: A plug-and-play framework designed to enhance logits-based detectors in black-box settings. Unlike prior surrogate-based methods, DALD aligns the surrogate model's probability distribution with that of the unknown target model. The aligned surrogate model significantly improves curvature-based or probability-divergence–based detection metrics. Following the original paper's setup: Fine-tuning the surrogate model, LLaMA2-7B, on the WildChat dataset.
- *DeTeCtive* [26]: DeTeCtive combines multi-level contrastive learning and multi-task auxiliary learning methods to enable fine-grained feature extraction that captures relationships between different text sources. In the experiments, We use SimCSE-RoBERTa-base [64] as the pretrain model. To preserve the original semantic space and prevent overfitting, we freeze the model's embedding layer during training. During testing, we adopt a KNN-based retrieval strategy using the validation set to automatically determine the optimal number of neighbors K within the range of 1 to 5, which is then used for predicting out-of-distribution data.

Introduction of Compared Commercial Detectors

The considered commercial detectors are as follows[8]:

[8] Official Pages: https://help.aliyun.com/document_detail/2979119.html, https://www.aiornot.com/, http://ruimei.ruijianai.com/, https://phrasly.ai/ai-detector, https://sapling.ai/, https://cloud.tencent.com/document/product/1124/118694, https://gptzero.me, https://www.zerogpt.com, https://originality.ai, https://www.turnitin.com, https://copyleaks.com

- *Aliyun:* The Content Moderation Service provided by Aliyun offers an AI-generated text detector, which analyzes the input content and returns judgment labels along with a confidence score. We truncate the input text if it exceeds the maximum length constraint of 600.
- *AIorNot:* AIorNot provides the continually updated detector trained on the latest generative models to provide a clear verdict and a confidence score for text provenance.
- *BlueEyes:* A text auditing service launched by Ruijian AI, which supports LLM-generated text detection in Chinese and English. We access the API equipped with the latest version.
- *Phrasly.ai:* A detection service that identifies patterns in language, sentence structure, and predictability. It claims a 99.8% accuracy rate in distinguishing human-written text from LLM-generated text from ChatGPT, Claude, and Gemini.
- *Sapling:* Sapling AI Content Detector leverages a Transformer-based system, similar in architecture to the models used to generate AI content. It is designed to detect text from various LLMs such as GPT-4, Claude, and Gemini. The official documentation reports an accuracy of 97% with a <3% false positive rate.
- *TencentCloud:* The AI-generated text detection service from Text Moderation System of Tencent's T-Sec series. It performs detection by analyzing both its formal features and content characteristics, and covers mainstream LLMs and diverse domains. The maximum input length is 10,000 characters.
- *GPTZero:* One of the most widely used AI detectors that utilizes a multi-step approach with seven components to analyze text, boasting high accuracy for major LLMs. Independent benchmarks often rank it as a leader in precision.
- *ZeroGPT:* A popular detection tool that employs sophisticated algorithms trained on millions of articles. It provides a percentage-based score to indicate varying levels of AI probability.
- *Originality.ai:* Its AI detector is claimed to be highly sensitive to the latest LLMs (like GPT-4.5 and Claude 3.7). We use the standard version.
- *Turnitin:* Turnitin integrated AI writing detection into its similarity report workflow. Its model is built on a transformer deep-learning architecture specifically trained to identify human writing versus generative AI.
- *Copyleaks:* It offers high-accuracy AI detection (over 99%) with a very low false-positive rate (0.2% as claimed) and is capable of detecting AI-generated text that has been heavily paraphrased.

References

[1] Kaddour J, Harris J, Mozes M, Bradley H, Raileanu R, McHardy R. Challenges and Applications of Large Language Models. https://arxiv.org/abs/2307.10169, 2023

[2] Koike R, Kaneko M, Okazaki N. OUTFOX: LLM-Generated Essay Detection Through In-Context Learning with Adversarially Generated Examples. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 21258–21266

[3] Chen C, Shu K. Combating Misinformation in the Age of LLMs: Opportunities and Challenges. AI Magazine, 2024, 45(3): 354–368

[4] Puccetti G, Rogers A, Alzetta C, Dell'Orletta F, Esuli A. AI 'News' Content Farms Are Easy to Make and Hard to Detect: A Case Study in Italian. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024, 15312–15338

[5] Guo B, Zhang X, Wang Z, Jiang M, Nie J, Ding Y, Yue J, Wu Y. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. https://arxiv.org/pdf/2301.07597, 2023

[6] Shi Y, Sheng Q, Cao J, Mi H, Hu B, Wang D. Ten words only still help: Improving black-box ai-generated text detection via proxy-guided efficient re-sampling. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. 2024, 494–502

[7] Hu E J, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W. LoRA: Low-Rank Adaptation of Large Language Models. In: The Tenth International Conference on Learning Representations. 2022

[8] Alibaba Cloud Community . Alibaba Open-sources Qwen3.6-35B-A3B; Wan2.7 Tops Design Arena. https://www.alibabacloud.com/blog/603042, 2026. Accessed: 2026-06-03

[9] Meta AI . The future of AI: Built with Llama. https://ai.meta.com/blog/future-of-ai-built-with-llama/, 2024. Accessed: 2026-06-03

[10] Wang P, Li L, Ren K, Jiang B, Zhang D, Qiu X. SeqXGPT: Sentence-Level AI-Generated Text Detection. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 1144–1156

[11] Li L, Wang P, Ren K, Sun T, Qiu X. Origin tracing and detecting of llms. https://arxiv.org/abs/2304.14072, 2023

[12] Bengio S, Vinyals O, Jaitly N, Shazeer N. Scheduled sampling for sequence prediction with recurrent neural networks. Advances in neural information processing systems, 2015, 28

[13] Bengio Y, Louradour J, Collobert R, Weston J. Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. 2009, 41–48

[14] Chen T, Kornblith S, Norouzi M, Hinton G. A Simple Framework for Contrastive Learning of Visual Representations. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 1597–1607

[15] Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, Bikel D, Blecher L, Ferrer C C, Chen M, Cucurull G, Esiobu D, Fernandes J, Fu J, Fu W, Fuller B, Gao C, Goswami V, Goyal N, Hartshorn A, Hosseini S, Hou R, Inan H, Kardas M, Kerkez V, Khabsa M, Kloumann I, Korenev A, Koura P S, Lachaux M A, Lavril T, Lee J, Liskovich D, Lu Y, Mao Y, Martinet X, Mihaylov T, Mishra P, Molybog I, Nie Y, Poulton A, Reizenstein J, Rungta R, Saladi K, Schelten A, Silva R, Smith E M, Subramanian R, Tan X E, Tang B, Taylor R, Williams A, Kuan J X, Xu P, Yan Z, Zarov I, Zhang Y, Fan A, Kambadur M, Narang S, Rodriguez A, Stojnic R, Edunov S, Scialom T. Llama 2: Open Foundation and Fine-Tuned Chat Models. https://arxiv.org/abs/2307.09288, 2023

[16] Gemma Team . Gemma: Open Models Based on Gemini Research and Technology. https://arxiv.org/abs/2403.08295, 2024

[17] Jiang A Q, Sablayrolles A, Mensch A, Bamford C, Chaplot D S, Casas d. l D, Bressand F, Lengyel G, Lample G, Saulnier L, Lavaud L R, Lachaux M A, Stock P, Scao T L, Lavril T, Wang T, Lacroix T, Sayed W E. Mistral 7B. https://arxiv.org/abs/2310.06825, 2023

[18] Bai J, Bai S, Chu Y, Cui Z, Dang K, Deng X, Fan Y, Ge W, Han Y, Huang F, Hui B, Ji L, Li M, Lin J, Lin R, Liu D, Liu G, Lu C, Lu K, Ma J, Men R, Ren X, Ren X, Tan C, Tan S, Tu J, Wang P, Wang S, Wang W, Wu S, Xu B, Xu J, Yang A, Yang H, Yang J, Yang S, Yao Y, Yu B, Yuan H, Yuan Z, Zhang J, Zhang X, Zhang Y, Zhang Z, Zhou C, Zhou J, Zhou X, Zhu T. Qwen Technical Report. https://arxiv.org/pdf/2309.16609, 2023

[19] Tufts B, Zhao X, Li L. A practical examination of AI-generated text

detectors for large language models. In: Findings of the Association for Computational Linguistics: NAACL 2025. April 2025, 4839–4856

[20] Chen Y, Kang H, Zhai V, Li L, Singh R, Raj B. Token Prediction as Implicit Classification to Identify LLM-Generated Text. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 13112–13120

[21] Yang X, Cheng W, Wu Y, Petzold L, Wang W, Chen H. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. In: International Conference on Learning Representations. 2024, 48572–48597

[22] Mitchell E, Lee Y, Khazatsky A, Manning C D, Finn C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 24950–24962

[23] Bao G, Zhao Y, Teng Z, Yang L, Zhang Y. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In: Kim B, Yue Y, Chaudhuri S, Fragkiadaki K, Khan M, Sun Y, eds, International Conference on Learning Representations. 2024, 24814–24836

[24] West A, Weng Y, Zhu M, Zhang L, Lin Z, Bao G, Zhang Y. Ai-generated text is non-stationary: Detection via temporal tomography. https://arxiv.org/abs/2508.01754, 2025

[25] Zeng C, Tang S, Yang X, Chen Y, Sun Y, Xu Z, Li Y, Chen H, Cheng W, Xu D. Dald: Improving logits-based detector without logits from black-box llms. Advances in Neural Information Processing Systems, 2024, 37: 54947–54973

[26] Guo X, Zhang S, He Y, Zhang T, Feng W, Huang H, Ma C. DeTeCtive: Detecting AI-generated Text via Multi-Level Contrastive Learning. In: Advances in Neural Information Processing Systems. 2024, 88320–88347

[27] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. https://arxiv.org/abs/1503.02531, 2015

[28] Li Z, Li X, Yang L, Zhao B, Song R, Luo L, Li J, Yang J. Curriculum temperature for knowledge distillation. In: Proceedings of the AAAI conference on artificial intelligence. 2023, 1504–1512

[29] Wu J, Zhan R, Wong D F, Yang S, Yang X, Yuan Y, Chao L S. DetectRL: Benchmarking LLM-Generated Text Detection in Real-World Scenarios. In: Advances in Neural Information Processing Systems. 2024, 100369–100401

[30] Nieth B, Gracheva M, Mahlberg M, Eskofier B, Salin E. How human-like are large language models? a register-aware linguistic evaluation framework. https://arxiv.org/abs/2605.23651, 2026

[31] Attar Y E, Dönmez E, Maurer M, Falenska A. A systematic analysis of linguistic features in ai-generated text detection across domains and models. https://arxiv.org/abs/2606.04177, 2026

[32] Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://arxiv.org/abs/1907.11692, 2019

[33] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu P J. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 2020, 21: 1–67

[34] Yu X, Chen K, Yang Q, Zhang W, Yu N. Text Fluoroscopy: Detecting LLM-Generated Text through Intrinsic Features. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024, 15838–15846

[35] Ji J, Guo J, Qiu W, Huang Z, Xu Y, Lu X, Jiang X, Li R, Li S. "I know myself better, but not really greatly": Using LLMs to Detect and Explain LLM-Generated Texts. https://arxiv.org/abs/2502.12743, 2025

[36] Li Y, Sheng Q, Yang Y, Zhang X, Cao J. From judgment to interference: Early stopping llm harmful outputs via streaming content monitoring. In: Belgrave D, Zhang C, Lin H, Pascanu R, Koniusz P, Ghassemi M, Chen N, eds, Advances in Neural Information Processing Systems. 2025, 54305–54333

[37] Zhu B, Yuan L, Cui G, Chen Y, Fu C, He B, Deng Y, Liu Z, Sun M, Gu M. 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. 2023, 7470–7483

[38] Hans A, Schwarzschild A, Cherepanova V, Kazemi H, Saha A, Goldblum M, Geiping J, Goldstein T. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In: Salakhutdinov R, Kolter Z, Heller K, Weller A, Oliver N, Scarlett J, Berkenkamp F, eds, Proceedings of the 41st International Conference on Machine Learning. 21–27 Jul 2024, 17519–17537

[39] Yu X, Qi Y, Chen K, Chen G, Yang X, Zhu P, Shang X, Zhang W, Yu N. Dpic: Decoupling prompt and intrinsic characteristics for llm generated text detection. In: Advances in Neural Information Processing Systems. 2024, 16194–16212

[40] Mao C, Vondrick C, Wang H, Yang J. Raidar: generative ai detection via rewriting. In: Kim B, Yue Y, Chaudhuri S, Fragkiadaki K, Khan M, Sun Y, eds, International Conference on Learning Representations. 2024, 6013–6030

[41] Chen J, Zhu X, Liu T, Chen Y, Chen X, Yuan Y, Leong C T, Li Z, Long T, Zhang L, Yan C, Mei G, Zhang J, Zhang L. Imitate before detect: Aligning machine stylistic preference for machine-revised text detection. In: Walsh T, Shah J, Kolter Z, eds, Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025. 2025, 23559–23567

[42] Zhu X, Ren Y, Fang F, Tan Q, Wang S, Cao Y. Dna-detectllm: Unveiling ai-generated text via a dna-inspired mutation-repair paradigm. In: Advances in Neural Information Processing Systems. 2025, 1745–1768

[43] Hao W, Li R, Zhao W, Yang J, Mao C. Learning to rewrite: Generalized LLM-generated text detection. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2025, 6421–6434

[44] Wu C, Cheung m Y, Han B, Lian D. Advancing machine-generated text detection from an easy to hard supervision perspective. https://arxiv.org/abs/2511.00988, 2025

[45] He X, Shen X, Chen Z, Backes M, Zhang Y. MGTBench: Benchmarking Machine-Generated Text Detection. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. 2024, 2251–2265

[46] Wang Y, Mansurov J, Ivanov P, Su J, Shelmanov A, Tsvigun A, Afzal O M, Mahmoud T, Puccetti G, Arnold T, others . M4GT-Bench: Evaluation Benchmark for Black-Box Machine-Generated Text Detection. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024, 3964–3992

[47] Liu A, Pan L, Lu Y, Li J, Hu X, Zhang X, Wen L, King I, Xiong H, Yu P. A Survey of Text Watermarking in the Era of Large Language Models. ACM Computing Surveys, 2024, 57(2): 1–36

[48] Liu A, Sheng Q, Hu X. Preventing and detecting misinformation generated by large language models. In: Proceedings of the 47th International

ACM SIGIR Conference on Research and Development in Information Retrieval. 2024, 3001–3004

[49] Xu X, Jia J, Yao Y, Liu Y, Li H. Robust multi-bit text watermark with LLM-based paraphrasers. In: Forty-second International Conference on Machine Learning. 2025

[50] Wu Q, Chandrasekaran V. Bypassing LLM Watermarks with Color-Aware Substitutions. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024, 8549–8581

[51] Pang Q, Hu S, Zheng W, Smith V. No Free Lunch in LLM Watermarking: Trade-offs in Watermarking Design Choices. In: Advances in Neural Information Processing Systems. 2024, 138756–138788

[52] Wu Z, Gong G, Zhu Q, Chen Y, Zhao R. Linear ensembles wash away watermarks: On the fragility of distributional perturbations in llms. https://arxiv.org/abs/2605.30501, 2026

[53] Kumarage T, Agrawal G, Sheth P, Moraffah R, Chadha A, Garland J, Liu H. A survey of ai-generated text forensic systems: Detection, attribution, and characterization. https://arxiv.org/abs/2403.01152, 2024

[54] Kumarage T, Liu H. Neural Authorship Attribution: Stylometric Analysis on Large Language Models . In: 2023 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. November 2023, 51–54

[55] Foley M, Rawat A, Lee T, Hou Y, Picco G, Zizzo G. Matching Pairs: Attributing Fine-Tuned Models to their Pre-Trained Large Language Models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). July 2023, 7423–7442

[56] Sarvazyan A M, González J Á, Rosso P, Franco-Salvador M. Supervised Machine-Generated Text Detectors: Family and Scale Matters. In: International Conference of the Cross-Language Evaluation Forum for European Languages. 2023, 121–132

[57] Thorat S, Yang T. Which LLMs are difficult to detect? a detailed analysis of potential factors contributing to difficulties in LLM text detection. In: Neurips Safe Generative AI Workshop 2024. 2024

[58] DeepSeek-AI . DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. https://arxiv.org/abs/2501.12948, 2025

[59] DeepSeek-AI . DeepSeek-V3 Technical Report. https://arxiv.org/abs/2412.19437, 2025

[60] Hu B, Sheng Q, Cao J, Shi Y, Li Y, Wang D, Qi P. Bad actor, good advisor: Exploring the role of large language models in fake news detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 22105–22113

[61] Sharma M, Tong M, Mu J, Wei J, Kruthoff J, Goodfriend S, Ong E, Peng A, Agarwal R, Anil C, Askell A, Bailey N, Benton J, Bluemke E, Bowman S R, Christiansen E, Cunningham H, Dau A, Gopal A, Gilson R, Graham L, Howard L, Kalra N, Lee T, Lin K, Lofgren P, Mosconi F, O'Hara C, Olsson C, Petrini L, Rajani S, Saxena N, Silverstein A, Singh T, Sumers T, Tang L, Troy K K, Weisser C, Zhong R, Zhou G, Leike J, Kaplan J, Perez E. Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hours of Red Teaming. https://arxiv.org/abs/2501.18837, 2025

[62] Niu C, Yancey K P, Liu R, Baig M B, Horie A K, Sharpnack J. Detecting LLM-assisted cheating on open-ended writing tasks on language proficiency tests. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track. November 2024, 940–953

[63] Black S, Gao L, Wang P, Leahy C, Biderman S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021

[64] Gao T, Yao X, Chen D. Simcse: Simple contrastive learning of sentence embeddings. In: Proceedings of the 2021 conference on empirical methods in natural language processing. 2021, 6894–6910

Yehan Yang is currently pursuing his Ph.D. degree at Institute of Computing Technology, Chinese Academy of Sciences. He received his B.E. degree from Beijing University of Posts and Telecommunications. His research interests include machine-generated text detection and LLM safety.

Yuhui SHI received his M.E. degree from Institute of Computing Technology, Chinese Academy of Sciences, and his B.E. degree from Beijing University of Posts and Telecommunications. His main research interests lie in machine-generated text detection and attribution.

Qiang SHENG is an associate professor at Institute of Computing Technology, Chinese Academy of Sciences, where he received his Ph.D. degree in 2023. His research interests include fake news detection, LLM-generated text detection, and LLM safety. He has published over 30 papers in conferences and journals, including ICML, NeurIPS, ACL, TKDE, etc.

Hao MI is currently pursuing his M.E. degree at Institute of Computing Technology, Chinese Academy of Sciences. He received his B.E. degree from Xi'an Jiaotong University. His research interest lies in LLM hallucination detection.

Beizhe HU is currently pursuing his Ph.D. degree at Institute of Computing Technology, Chinese Academy of Sciences. He received his B.E. degree from Xidian University. His research interest is fake news detection in the ear of large language models.

Chaoxi XU is currently pursuing his Ph.D. degree at Institute of Computing Technology, Chinese Academy of Sciences. He received his M.S. degree from Renmin University of China. His research interest lies in LLM-generated text detection.

Juan CAO is a professor at Institute of Computing Technology, Chinese Academy of Sciences, where she received her Ph.D. degree in 2008. Her research interests include AI-generated content safety and multimedia technology. She has over 100 publications in international journals and conferences, including TKDE, NeurIPS, ICLR, ACL, etc.