

CEDAR: Creator–Editor Disentanglement and Recalibration for LLM-Generated Chinese Text Detection

Bangyuan Wang ^{*}, Boyuan Song, and Jiacheng Ye

School of Computer Science, Beijing Institute of Technology

wangbangyuan@bit.edu.cn

1807796965@qq.com

1120241876@bit.edu.cn

Abstract. Detecting LLM-refined text and generalizing to OOD (out-of-distribution) scenarios remain critical challenges in machine-generated text detection. This paper describes our submission to NLPCC 2026 Task 6 on three-class Chinese text detection. We present CEDAR, a Creator–Editor disentanglement framework that factorizes the task along Creator and Editor axes. To enhance OOD robustness, we incorporate content-controlled contrastive learning and generator-adversarial training to align style representations, alongside removal of surface shortcuts and Editor-bias calibration using a single scalar. Without parameter stacking or model ensembling, a single CEDAR model with a RoBERTa-large backbone achieves a macro-F1 of 0.7299 on the Phase 2 blind test set, ranking in the top five on the official leaderboard. Ablation studies demonstrate that these gains stem from architectural alignment and targeted fixes for out-of-distribution shift rather than from model scale or voting.

Keywords: LLM-generated text detection · Chinese NLP · Creator–Editor framework · Out-of-distribution robustness

1 Introduction

Large language models now write text that is difficult to distinguish from human prose, which has made reliable detection of machine-generated content a pressing problem. Early detection work concentrated on the binary case: given a passage, decide whether it was written by a human or produced by a model. While this formulation remains useful, it no longer matches many practical scenarios in which a human draft is polished, expanded, or condensed by an LLM. Such text, initiated by a human and refined by an LLM, sits between the two binary extremes: it carries human content and intent, yet it also bears the stylistic signature of a model editor. Treating it as either purely human or purely machine therefore misrepresents the generative process and complicates both regulation and attribution.

^{*} Corresponding author.

The NLPCC 2026 Task 6 shared task formalizes this richer setting as a three-way Chinese text classification problem. Each input must be labeled as human-written text (HWT), LLM-generated text (LGT), or human-written text that has been refined by an LLM (HLT). The official evaluation metric is macro-averaged F1, and the test data are deliberately out-of-distribution: they come from different domains and are produced by different generators than the training set. A detector must therefore generalize across both topic and generator rather than rely on surface cues or fingerprints of particular models.

This requirement exposes the limitations of existing approaches. Training-free detectors such as Fast-DetectGPT and Binoculars perform well on the binary distinction between human and machine text, but they are not designed to recognize the intermediate HLT class and their reliability drops when the generator is unknown or the text has been edited. Supervised flat classifiers can learn the three labels, yet they easily overfit to shortcuts tied to particular datasets, including half-width punctuation and crawl residue in the training HWT. These shortcuts do not survive the train-test shift. RACE [10] offers a more structured alternative by modeling Creator and Editor roles separately, but its original four-class design and its reliance on English Rhetorical Structure Theory do not transfer cleanly to the Chinese three-class setting.

We address these gaps with CEDAR (Creator–Editor Disentanglement and Recalibration). The core idea is to factorize the three task labels into two binary axes, Creator (H/L) and Editor (H/L), and to train a pretrained Chinese encoder, instantiated with RoBERTa-large in our submitted system, to predict both axes jointly. This decomposition is combined with four complementary measures: a surface normalization procedure that preserves content while removing shortcuts based on punctuation and fragments from web crawls; content-controlled supervised contrastive learning and a generator-adversarial head that exploit the parallel HWT/HLT/LGT triplets in the training data; random-window training and windowed inference to handle long documents; and a single-scalar Editor-bias calibration that corrects a systematic OOD shift without fitting the test set.

On the NLPCC 2026 Task 6 leaderboard our submission achieves a macro-F1 of 0.7299 on the Phase 2 blind test set, placing in the top five. The gains are not due to any single trick: ablations show that the Creator–Editor head alone closes most of the gap against a flat classifier, while the remaining components each address a distinct failure mode of out-of-distribution generalization. Backbone-transfer tests further instantiate CEDAR with two larger Chinese encoders without backbone-specific tuning (§4.4).

2 Related Work

Detecting LLM-generated text has progressed from output-probability and surface statistics to learned classifiers and, more recently, internal representations and model-behavior signals.

Training-free methods exploit the higher likelihood of generated text under the generating model. Early work used perplexity and entropy thresholds [14,4]; later methods such as DetectGPT [12], Fast-DetectGPT [1], and Binoculars [6] improved robustness through perturbation or discrepancies across models. These approaches need no data from the target task, but their reliability drops when the generator is unknown or the text has been edited.

Recent methods draw on evidence beyond output probabilities. Using a surrogate model, RepreGuard [2] projects internal activations onto a learned direction that separates human and generated text. Learning2Rewrite [7] instead exploits the observation that LLMs rewrite generated text less than human text, using the resulting edit-distance signal to improve open-world generalization.

Training-based methods instead adapt pretrained language models using human and machine text [18,5]. Chinese detectors often use RoBERTa variants with whole word masking, which capture stylistic and syntactic cues that perplexity misses. Supervised detectors usually outperform training-free alternatives in-domain, yet they risk overfitting to shortcuts tied to individual datasets such as punctuation patterns unless carefully regularized.

A distinct direction models the generative process rather than flattening detection into a single classification. DETree [8] aligns text representations with a hierarchy of human–AI collaboration processes. RACE [10] introduces a Creator–Editor framework in which each sample arises from a creator and an editor, and the detector infers both roles jointly. Our work builds on this idea: we cast the three-way distinction as a Creator–Editor prediction problem and augment the framework with shortcut removal, stochastic windowing, contrastive supervision informed by content, and a generator-adversarial head to improve generalization across domains and generators.

3 Method

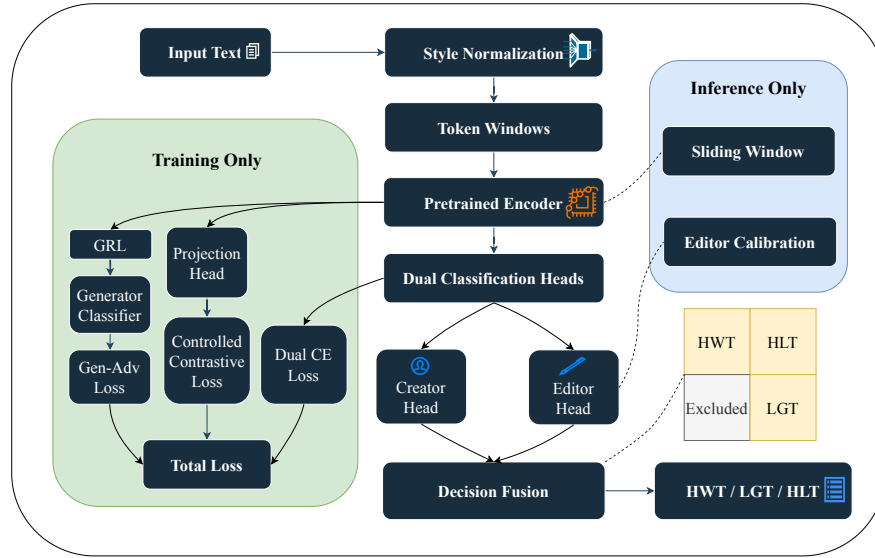
We refer to our approach as CEDAR (Creator–Editor Disentanglement and Recalibration). It is motivated by the task’s natural Creator–Editor structure and the parallel HWT/HLT/LGT triplet available for every training source. CEDAR decomposes the task along the Creator and Editor axes (§3.3), removes the surface shortcut (§3.2), adds content-controlled and generator-invariant training objectives (§3.4), handles long documents via random windows (§3.5), and recalibrates the Editor axis with a single scalar (§3.6).

3.1 Problem Formulation and Notation

We use the label set $\mathcal{Y} = \{\text{HWT}, \text{LGT}, \text{HLT}\}$ (human-written, LLM-generated, LLM-refined text) introduced in §1, with macro-averaged F1 on out-of-distribution test data. Following the creator–editor formulation of [10], we factorize each sample into two binary latent axes, creator $C \in \{H, L\}$ and editor $E \in \{H, L\}$, and read the three task labels as the three valid combinations of these two axes:

Table 1. Mapping from the three task labels to the Creator–Editor axes. The combination (L, H) is excluded by construction.

Task label	Creator	Editor
HWT	H	H
LGT	L	L
HLT	H	L

**Fig. 1.** CEDAR architecture: pretrained encoder, dual Creator/Editor heads, preprocessing, auxiliary training objectives, and Editor-bias calibration.

If the three-class label is $y \in \{0, 1, 2\}$ in the order HWT/LGT/HLT, the corresponding targets are $C[y] = (0, 1, 0)$ and $E[y] = (0, 1, 1)$. The rest of the paper predicts both axes jointly and recombines them with the product rule; this is what makes the decomposition useful in practice, since it gives both an interpretable binary view and a single three-way decision from the same four logits.

3.2 Preprocessing: Closing the Surface Artifact Shortcut

The first failure mode is a shortcut based on surface artifacts: training HWT texts carry residue from web crawls, especially half-width punctuation in Chinese context and short repeated boilerplate fragments, while the official test set is substantially cleaner. A detector trained on the raw data can therefore exploit a trivial rule of the form “messy surface implies HWT”, which does not survive the train–test shift.

We close this shortcut with a surface normalization procedure that preserves content and is applied uniformly to all three classes before tokenization. One pass collapses substrings of 2–12 characters that occur at least three times, targeting crawl boilerplate. Another pass uses local context to convert half-width punctuation to full-width only when adjacent to a CJK character, preserving Western numerals and abbreviations. The two passes are applied identically to HWT, LGT, and HLT; the normalization flag is persisted with the checkpoint so training and inference agree.

3.3 Creator–Editor Dual-Axis Modeling

The base detector is a pretrained bidirectional encoder (Chinese RoBERTa-large in our submitted system) followed by a two-layer classification head that emits four logits

$$\ell = [\ell_H^C, \ell_L^C, \ell_H^E, \ell_L^E], \quad (1)$$

with the first two entries for the Creator axis and the last two for the Editor axis. Let y denote the three-way label. Training minimizes the sum of two binary cross-entropies,

$$\mathcal{L}_{ce} = \text{CE}(\ell^C, C[y]) + \text{CE}(\ell^E, E[y]), \quad (2)$$

where the two binary cross-entropies enter with equal weight. Compared to a flat three-way classifier, this form lets the model learn two semantically meaningful subproblems at once: who created the content and whether the content has been edited by an LLM.

At inference, the two binary distributions $\sigma^C = \text{softmax}(\ell^C)$ and $\sigma^E = \text{softmax}(\ell^E)$ are recombined by the product rule on the three valid cells of Table 1:

$$p_{\text{HWT}} = \sigma_H^C \sigma_H^E, \quad p_{\text{LGT}} = \sigma_L^C \sigma_L^E, \quad p_{\text{HLT}} = \sigma_H^C \sigma_L^E. \quad (3)$$

The product rule also assigns mass $\sigma_L^C \sigma_H^E$ to the invalid (L, H) cell; we discard that mass and renormalize the three valid scores to sum to 1 before decoding by an argmax. This decoding rule keeps training and inference aligned: the same four logits support both an interpretable binary decomposition and the final three-way decision.

3.4 Training with Content-Controlled Auxiliary Objectives

Every training source is available as a parallel HWT/HLT/LGT triplet, so a minibatch can be assembled from complete triples rather than from independent samples. The two auxiliary objectives below both exploit this structure: they only make sense when content is held fixed across the positives and negatives of each anchor.

Content-controlled supervised contrastive learning The first auxiliary objective is supervised contrastive learning [9]. We use a small projection head to map the pooled encoder representation h to 128 dimensions, normalize the result to obtain z , and apply the standard SupCon loss with positives defined by the binary label on a chosen axis. Because the minibatch contains the full triplet for each source, an anchor’s positives and negatives on the Creator axis (or the Editor axis) are drawn from texts of the same content; the contrastive gradient therefore compares samples that differ only in the create or edit operation, not in topic, domain, or lexical surface.

We apply the objective on both axes:

$$\mathcal{L}_{\text{con}} = \mathcal{L}_{\text{SupCon}}(z, C[y]) + \mathcal{L}_{\text{SupCon}}(z, E[y]), \quad (4)$$

with temperature τ (set to 0.1 in all experiments). The projection head is used only during training. The loss is complementary to the classifier: the classification head learns the final Creator–Editor decision, while the contrastive head shapes the representation so that samples with the same axis label cluster across sources, encoding the edit operation rather than domain or style.

Generator-adversarial head The second auxiliary objective removes fingerprints specific to a generator from the shared representation. Although the task labels do not identify the generator, the training metadata identifies it for samples generated by machines. We attach a classifier for the four generators to the pooled representation, but feed the encoder through a gradient reversal layer [3] so that the encoder is trained to confuse the generator predictor rather than help it. Formally, with generator label g_i and reversal strength λ ,

$$\mathcal{L}_{\text{gen}} = \text{CE}(g_\phi(\text{GRL}_\lambda(h)), g_i), \quad (5)$$

restricted to samples generated or refined by machines (HWT has no generator identity). The shared encoder is thereby pushed toward generator-invariant features, which is important for the test setting with unseen generators and complements the content-controlled objective above.

3.5 Robustness to Long Text: Random-Window Augmentation and Windowed Inference

The test documents are much longer than the encoder input budget (the median in our data is well above the limit of 512 tokens), and a plain prefix truncation would discard a large fraction of the document. This is especially harmful for HWT, whose human-written prefix often contains exactly the source cues that the refined continuation later removes. A model that only sees the beginning of the text is therefore forced to classify on a systematically biased view.

We address this with two complementary changes. Let n denote the number of characters in a preprocessed training text. With probability 0.5, we uniformly sample an integer window length L from $\min(256, n)$ to n , and then uniformly

sample its start from all valid positions. The resulting substring is retained before tokenization and 512-token truncation. During inference, the tokenizer partitions the full text into windows of at most 512 tokens, with adjacent windows sharing 64 tokens. The four logits are averaged across windows before applying the product rule. Averaging logits, rather than probabilities, is important because probability averaging overweights already confident windows, whereas logit averaging preserves the relative evidence accumulated across the full document. The two changes are designed together: random-window training makes any segment supplied by windowed inference part of the encoder’s training distribution.

3.6 Test-Time Editor-Bias Calibration

Under OOD shift, prior and label distributions can differ from those seen at training time, and the standard correction is a per-axis logit shift [13,11], the same family as temperature or Platt scaling on a single binary axis. Because the dual-head decomposition emits two independent binary classifiers, the natural correction form under prior/label shift is a per-axis logit shift on one or both axes. To keep calibration deliberately constrained, we apply a single shift to the Editor axis and leave the Creator axis untouched; the Editor bias b is the only free parameter. Let $D = \log(p_H^E/p_L^E)$ and $D' = D + b$, with $\sigma(\cdot)$ the sigmoid. The calibrated three-class probabilities are

$$\begin{aligned} p_{\text{HWT}} &= \sigma_H^C \cdot \sigma(D'), \\ p_{\text{HLT}} &= \sigma_H^C \cdot (1 - \sigma(D')), \\ p_{\text{LGT}} &= \sigma_L^C \cdot (1 - \sigma(D')). \end{aligned} \tag{6}$$

These three scores do not sum to 1 because the invalid (L, H) cell still carries mass $\sigma_L^C \sigma(D')$; we discard it and renormalize the three valid cells, exactly as in the uncalibrated product rule of §3.3. The Creator axis remains unchanged throughout.

The bias has one free parameter, fit on a labeled hold-out of the training set; no test-set information is used. After selection the same scalar is reused unchanged for both Phase 1 and Phase 2.

The correction is a per-sample rule on the Editor log-odds, so it carries no target class ratio and does not use batch composition, test-set size, or empirical priors. It leaves the encoder and auxiliary objectives untouched; its effect is reported in Table 2.

4 Experiments

4.1 Experimental Setup

Datasets Both the training and test sets are the official ones released by the NLPCC 2026 Task 6 organizers, sourced from the Chinese portions of CUDRT [15] (training) and DetectRL-X [16] (test, the multilingual extension of DetectRL [17]). The training set contains 19,634 aligned HWT / LGT / HLT triples,

produced by four generators (Qwen, ChatGLM, Baichuan, GPT-4) over two domains (News, Thesis) under a protocol that generates continuations from prefixes. The test set is constructed using summarization followed by generation for LGT, three editing operations (polish, expand, condense) for HLT, and four disjoint generators (Deepseek-V3, Gemini-2.5-flash, GPT-4o, Qwen-Max). Beyond the shift in generators, the test set is no longer confined to the News and Thesis domains of the training set, making it out-of-distribution along both axes.

Two distributional properties drive our design. First, the median test document length is 1,464 characters and the 90th percentile is 2,256 characters, well beyond our encoder’s budget of 512 tokens, so all training and inference use a windowed protocol. Second, the test set undergoes much more aggressive surface normalization than the training set, creating a train–test surface shift that motivates the de-shortcutting step of §3.2.

Implementation We further train the `hf1/chinese-roberta-wwm-ext-large` encoder with AdamW for 2 epochs in fp16. We use an encoder learning rate of 1×10^{-5} and a $5\times$ larger head+auxiliary rate of 5×10^{-5} , weight decay 0.01, a 6% linear warmup, and an effective batch of 12 training triples (36 aligned samples). The two auxiliary-loss weights $\lambda_c=0.2$, $\lambda_g=0.8$ are selected on the leave-one-generator (Baichuan) OOD validation slice, with the weight sweep reported in Table 5; the single `editor_bias` $b=2.5$ is selected on a separate leave-one-domain (Thesis) hold-out, with the bias sweep in Figure 2; the same values are reused unchanged for both Phase 1 and Phase 2. All preprocessing and augmentation flags are persisted with the checkpoint, so the same settings are used for training and inference, as specified in §3.2 and §3.5. All models are trained on a single NVIDIA RTX 5090. Code and configs are released at https://github.com/WomderBoY/llm_generated_text_detection.

Evaluation The official metric is macro-averaged F1. Unless otherwise noted, all scores in the experiments section are from the Phase 1 OOD test set; the Phase 2 blind test result is reported in the introduction. During development, the regular training validation is a stratified 10% slice of the training set spanning both domains. The auxiliary-loss weights λ_c , λ_g are selected on a leave-one-generator (Baichuan) OOD slice, while the Editor bias b of §3.6 is selected on a separate leave-one-domain (Thesis) hold-out totaling $\sim 4,500$ records after capping within each class (the full Thesis hold-out has $\sim 7,634$ triples). Both slices are labeled, are taken from the training set, and are not the official test set. No official test text, label, prediction, or aggregate statistic is used to select a hyperparameter, an auxiliary-loss weight, or the `editor_bias` b . The pipeline is fully per-sample at inference time: a text is encoded once into four logits, b is added to the Editor logit, the product rule of §3.3 yields three scores, and the argmax is the predicted label. No batch or transductive statistic on the test set enters the decision.

4.2 Main Results and Ablation Study

Table 2 reports the macro-F1 of the proposed CEDAR pipeline on the Phase 1 OOD test set, progressively adding one component at a time. Each row keeps every earlier component; the Δ column is the increment introduced by the row’s component. Row 1 is the internal starting point that all subsequent rows build on, not a cross-system baseline: it shares the encoder, optimizer, and training corpus with the full pipeline, and differs from row 2 only in the output head. Cross-system comparisons against zero-shot detectors and against the original RACE are deferred to §4.3.

Table 2. Ablation of CEDAR on the Phase 1 OOD test set. Each row adds one component; Δ is the macro-F1 increment.

# Configuration	macro-F1	Δ
1 RoBERTa-large + flat 3-class head	0.4158	—
2 + Creator–Editor dual head (§3.3)	0.5946	+0.1788
3 + Surface de-shortcutting (§3.2)	0.6360	+0.0414
4 + Random-window training + windowed inference (§3.5)	0.6494	+0.0134
5a + Content-controlled SupCon only (§3.4)	0.6778	+0.0284
5b + Generator-adversarial only (§3.4)	0.6680	+0.0186
6 + SupCon $\cup$ Gen-Adv (§3.4, full)	0.6940	+0.0162
7 + Editor-bias calibration (§3.6)	0.7363	+0.0423

The first cluster of gains comes from structure and data hygiene. Replacing the flat three-class head with the Creator–Editor dual head alone closes a 0.1788 gap on the out-of-distribution slice, the largest improvement contributed by any component in the table. This confirms that the axis decomposition of §3.3, rather than the auxiliary losses, is the dominant factor: the model receives exactly the two binary supervision signals the method was designed around, and the product rule accounts for most of the gain. Adding surface normalization on top contributes another 0.0414, a small but steady increment, in line with the train–test surface shift discussed in §4.1. Windowed training and windowed inference together add 0.0134: the gain is modest in isolation, but the same flag is required for the regime of long documents that dominates the test slice, and removing it would make a fair comparison against the Editor-bias step impossible.

The second cluster concerns the two auxiliary objectives of §3.4. Rows 5a and 5b confirm that each is useful in isolation, contributing 0.0284 and 0.0186 respectively (each measured against the 0.6494 baseline in row 4). Stacking the second objective on top of the first, row 6 reaches 0.6940, improving by 0.0162 over row 5a. This is an additive rather than redundant effect. The two objectives are not interchangeable: content-controlled contrast pulls the HLT embedding toward HWT on the Creator axis and toward LGT on the Editor axis, while generator-adversarial pressure removes the dependency on the specific source

Table 3. Cross-system comparison on the Phase 1 OOD test set (Codabench). ‡ denotes our implementation for this paper.

Method	Arch	macro-F1
Binoculars [6]	statistical	0.3772
Fast-DetectGPT [1]	statistical	0.4750
RACE (our implementation) [‡] [10]	—	0.3746
RoBERTa-large + flat 3-class head	neural (encoder)	0.4158
Qwen-3.5-9B + LoRA + flat 3-class head	neural (decoder)	0.3202
CEDAR (ours)	neural (encoder)	0.7363

model. Neither direction suffices on its own; with both present, the Editor head no longer carries a single generator’s signature, and the Creator head no longer uses content as a covert channel for the source identity.

The third cluster is the Editor-bias calibration of §3.6. Row 7 gives the largest gain after training, improving on row 6 by 0.0423 with a single scalar. The magnitude is consistent with what one expects for a constant shift in the logits on a single binary axis: small enough that a single parameter cannot memorize 3,600 test points, and large enough to indicate that the bias captures a systematic offset on the Editor log-odds rather than per-sample noise. We examine this step further through the bias sweep in Figure 2.

4.3 Cross-System Baselines

Table 3 compares CEDAR against zero-shot anchors, our implementation of RACE [10], and two supervised references using flat classifiers, which isolate the effect of using an encoder rather than a decoder.

The zero-shot anchors fall into the 0.37–0.48 range, far below the supervised regime, so CEDAR is built on supervised labels rather than a likelihood anchor. Our implementation of RACE scores only 0.3746 because its four-class design (which separates “LLM-Polished HWT” from “Humanized LLM Text”) maps poorly to the three-class task, and its English RST feature engineering has no validated Chinese counterpart. The gap between the encoder and decoder is consistent with the diagnosis of surface shortcuts: the Qwen-3.5-9B decoder with a flat head scores above 0.95 on the in-domain validation set but only 0.3202 on the OOD test set, well below RoBERTa-large’s 0.4158 despite a $27\times$ parameter advantage. This collapse reflects overfitting to surface artifacts in the training set. We report this as a sanity check rather than a controlled ablation.

4.4 Backbone Transfer

To test backbone portability, we instantiate CEDAR with two Chinese Erlangshen encoders: DeBERTa-v2-710M and MegatronBERT-1.3B [19]. Without backbone-specific tuning, both use the RoBERTa setting $(\lambda_c, \lambda_g) = (0.2, 0.8)$, a common one-epoch recipe, and zero inference biases.

Table 4. Untuned backbone transfer of CEDAR without axis-bias calibration.

Encoder	Phase 1 macro-F1	Phase 2 macro-F1
DeBERTa-v2-710M	0.6915	0.6933
MegatronBERT-1.3B	0.6954	0.7154

Both Phase 1 scores are within 0.0025 of the uncalibrated RoBERTa-large result (0.6940) in Table 2, while Phase 2 reaches 0.6933 and 0.7154. These untuned results demonstrate portability, not a ranking of encoders.

4.5 Hyperparameter and Calibration Analysis

On the leave-one-generator (Baichuan) OOD slice, 6 of 8 settings in Table 5 cluster in $[0.7346, 0.7475]$. The selected $(\lambda_c, \lambda_g) = (0.2, 0.8)$ is best at 0.7592, but the analysis is not tied to this precise choice.

Table 5. Auxiliary-loss weight sensitivity on the leave-one-generator (Baichuan) OOD validation slice.

λ_c	λ_g	val macro-F1
0.2	0.2	0.7421
0.2	0.5	0.7174
0.2	0.8	0.7592
0.5	0.2	0.7475
0.5	0.5	0.7346
0.5	0.8	0.7363
0.8	0.2	0.7395
0.8	0.8	0.7391

Figure 2 sweeps b on a separate leave-one-domain (Thesis) hold-out (so the uncalibrated 0.6230 at $b=0$ is on a different OOD axis from Table 5). The curve has a single peak at $b = 2.5$ with a flat plateau over $b \in [2.0, 3.0]$; because the curve is not monotonic, it rules out a correction that varies strictly in one direction. The same $b = 2.5$ transfers cleanly to Phase 1 and Phase 2.

5 Conclusion

We presented CEDAR, a Creator–Editor disentanglement approach for the three-way Chinese detection task in NLPCC 2026 Task 6. By factorizing the labels into Creator and Editor axes and combining surface de-shortcutting, content-controlled contrastive learning, generator-adversarial training, random-window augmentation, and a single-scalar Editor-bias calibration, our submission achieves a macro-F1 of 0.7299 on the Phase 2 blind test set.

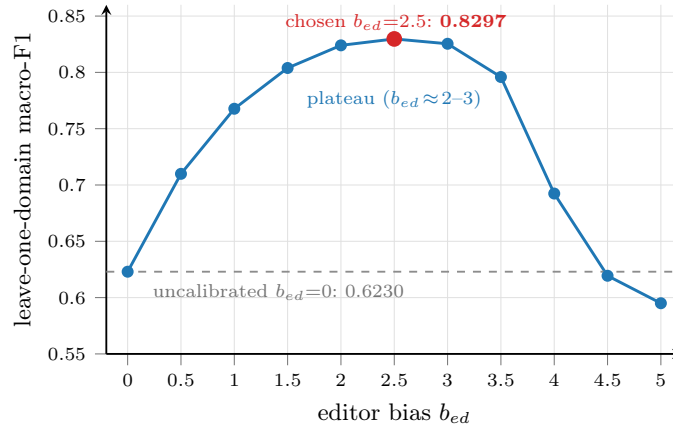


Fig. 2. Editor-bias sensitivity on the leave-one-domain (Thesis) validation slice (§3.6).

The result is obtained with a single RoBERTa-large backbone and no model ensemble, so the gains are not attributable to parameter stacking or voting. Instead, the ablations show that the largest improvements come from aligning the model architecture with the task structure: the dual-axis head alone closes most of the gap against a flat classifier, while the remaining components each address a distinct out-of-distribution failure mode. The success of CEDAR therefore lies in a sequence of targeted fixes for the train-test shift rather than in scale or model averaging.

Acknowledgments. We thank Dr. Xin Xin for teaching the Knowledge Engineering course, through which we studied deep learning and natural language processing, and for encouraging us to participate in the NLPCC shared task to gain practical experience. This participation directly motivated the work reported in this paper.

References

1. 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: Proc. ICLR (2024)
2. Chen, X., Wu, J., Yang, S., Zhan, R., Wu, Z., Luo, Z., et al.: RepreGuard: Detecting LLM-generated text by revealing hidden representation patterns. Transactions of the Association for Computational Linguistics **13**, 1812–1831 (2025). <https://doi.org/10.1162/tacl.a.61>
3. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., et al.: Domain-adversarial training of neural networks. Journal of Machine Learning Research **17**(59), 1–35 (2016)
4. Gehrmann, S., Strobel, H., Rush, A.M.: GLTR: Statistical detection and visualization of generated text. In: Proc. ACL: System Demonstrations. pp. 111–116 (2019). <https://doi.org/10.18653/v1/P19-3019>

5. Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., et al.: How close is ChatGPT to human experts? comparison corpus, evaluation, and detection. arXiv preprint arXiv:2301.07597 (2023), <https://arxiv.org/abs/2301.07597>
6. Hans, A., Schwarzschild, A., Cherepanova, V., Kazemi, H., Saha, A., Goldblum, M., et al.: Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In: Proc. ICML. Proceedings of Machine Learning Research, vol. 235, pp. 17519–17537 (2024)
7. Hao, W., Li, R., Zhao, W., Yang, J., Mao, C.: Learning to rewrite: Generalized LLM-generated text detection. In: Proc. ACL. pp. 6421–6434 (2025). <https://doi.org/10.18653/v1/2025.acl-long.322>
8. He, Y., Zhang, S., Cao, Y., Ma, L., Luo, P.: DETree: DETecting human–AI collaborative texts via tree-structured hierarchical representation learning. In: Proc. NeurIPS. vol. 38 (2025)
9. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., et al.: Supervised contrastive learning. In: Proc. NeurIPS. vol. 33, pp. 18661–18673 (2020)
10. Li, Y., Sheng, Q., Wang, Z., Yang, Y., Wang, D., Cao, J.: Beyond the final actor: Modeling the dual roles of creator and editor for fine-grained LLM-generated text detection. In: Proc. ACL. pp. 5188–5203 (2026). <https://doi.org/10.18653/v1/2026.acl-long.235>
11. Menon, A.K., Jayasumana, S., Rawat, A.S., Jain, H., Veit, A., Kumar, S.: Long-tail learning via logit adjustment. In: Proc. ICLR (2021)
12. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C.D., Finn, C.: DetectGPT: Zero-shot machine-generated text detection using probability curvature. In: Proc. ICML. Proceedings of Machine Learning Research, vol. 202, pp. 24950–24962 (2023)
13. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computation **14**(1), 21–41 (2002). <https://doi.org/10.1162/089976602753284446>
14. Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., et al.: Release strategies and the social impacts of language models. arXiv preprint arXiv:1908.09203 (2019), <https://arxiv.org/abs/1908.09203>
15. Tao, Z., Chen, Y., Xi, D., Li, Z., Xu, W.: Toward reliable detection of LLM-generated texts: A comprehensive evaluation framework with CUDRT. ACM Transactions on Intelligent Systems and Technology **17**(2), 1–35 (2026). <https://doi.org/10.1145/3779427>
16. Wu, J., Liu, Y., Zhu, C., Zhang, H., Wu, Z., Shi, T., et al.: DetectRL-X: Towards reliable multilingual and real-world LLM-generated text detection. In: Proc. ACL. pp. 38247–38294 (2026). <https://doi.org/10.18653/v1/2026.acl-long.1773>
17. Wu, J., Zhan, R., Wong, D.F., Yang, S., Yang, X., Yuan, Y., et al.: DetectRL: Benchmarking LLM-generated text detection in real-world scenarios. In: Proc. NeurIPS. vol. 37, pp. 100369–100401 (2024). <https://doi.org/10.52202/079017-3186>
18. Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., et al.: Defending against neural fake news. In: Proc. NeurIPS. vol. 32, pp. 9054–9065 (2019)
19. Zhang, J., Gan, R., Wang, J., Zhang, Y., Zhang, L., Yang, P., et al.: Fengshenbang 1.0: Being the foundation of Chinese cognitive intelligence. CoRR **abs/2209.02970** (2022)