UIT-AMMC at SemEval-2026 Task 13: Exploiting Structural Formatting Signatures for Robust AI-Generated Code Detection

Pham Quoc Cuong[1,2], **Nguyen Ngoc Minh**[1,2], **Le Quang Minh**[1,2]
Nguyen Hoai Nguyet An[1,2], **Nguyen Trong Chinh**[1,2]
[1]University of Information Technology, Ho Chi Minh City, Vietnam
[2]Vietnam National University Ho Chi Minh City, Vietnam
{cuongpq.20, minhnn.20}@grad.uit.edu.vn
{minhlq.20, annhn.20}@grad.uit.edu.vn, chinhnt@uit.edu.vn

Abstract

We propose the **Structure-Aware Contrastive Cascade**, an architecture designed to mitigate *syntactic convergence* between human-written and AI-generated code. Our approach identifies **Structural Formatting Signatures**—characterized by specific density distributions in the code's physical layout—as a robust forensic marker for AI-generated content. Leveraging the 500K sample algorithmic training corpus, we fine-tune a **Qwen-2.5-Coder 14B** model using QLoRA. To maintain reliability across unseen languages (Go, PHP, C#, C, JS) and functional domains (Research, Production), the system fuses contrastive generative likelihoods with structural priors. For ambiguous cases, we apply a multi-agent refinement step inspired by **Agentic Context Engineering (ACE)**, a framework that treats contexts as evolving playbooks to accumulate, refine, and organize strategies. Our system achieved a **Macro F1 score of 0.802**, ranking **3rd on the official leaderboard**, demonstrating that structural heuristics can effectively complement large-scale reasoning for robust, cross-domain code forensics.

1 Introduction

The democratization of Large Language Models (LLMs) in software engineering necessitates robust forensics to distinguish human-written from AI-generated (Becker et al., 2023). However, detection is hindered by *syntactic convergence*: a phenomenon where compiler constraints drive both human-written and AI-generated solutions into a narrow functional manifold. This renders traditional semantic classifiers and zero-shot metrics increasingly ineffective (Guo et al., 2025; Li et al., 2025), especially for out-of-distribution (OOD) generalization across unseen languages and functional domains.

In this work for SemEval-2026 Task 13 Subtask A (Orel et al., 2026b), we evaluate detection paradigms using the official 500K-sample algorithmic corpus (Orel et al., 2026a). We identify that post-training alignment inadvertently introduces systematic structural presentation regularities optimized for code readability. We formalize this as a forensic marker characterized by specific density distributions in the code's physical structure, which remain informative even when semantic signals are blurred. Our experiments show that established methods—including neural stylometry (Li et al., 2025) and fast-inference contrastive models (Fu et al., 2025)—struggle with OOD scenarios due to the low-entropy nature of source code.

We propose the Structure-Aware Contrastive Cascade, a multi-view architecture integrating structural priors with generative reasoning. The pipeline involves four phases: (1) extracting structural heuristics such as the empty-line ratio (R_{void}) to summarize layout regularities; (2) feature-aware QLoRA fine-tuning of a Qwen-2.5-Coder 14B model (Dettmers et al., 2023) so the model conditions on both code and forensic cues; (3) contrastive inference that compares the evidence for human and AI authorship; and (4) an optional debate-based refinement stage, inspired by Agentic Context Engineering (ACE) (Zhang et al., 2025), that is invoked only for uncertain cases.

Our system achieved a Macro F1 of 0.802, ranking 3rd among 82 teams. Our primary contributions are:

- Identifying structural formatting signatures as robust, language-agnostic markers for OOD detection in AI-generated code.
- Proposing the Structure-Aware Contrastive Cascade to fuse structural heuristics with code-optimized LLM reasoning. This framework integrates parameter-efficient fine-tuning with explicit physical-layout cues to bridge deterministic formatting metrics and generative likelihood analysis.

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 409–417

- Analyzing failed paradigms in the face of syntactic convergence and distribution shift.

2 Background and Related Work

2.1 Task Overview

SemEval-2026 Task 13 Subtask A studies binary AI-generated code detection under realistic distribution shift (Orel et al., 2026b). The objective is to classify each snippet as Human or AI-generated. The task is intentionally difficult because models are trained on algorithmic code in a limited set of languages, but evaluation extends to unseen languages and to functional domains whose formatting and coding conventions differ from the training distribution.

The official corpus contains approximately 500K labeled training samples, with 238K human-written and 262K AI-generated examples (Orel et al., 2026a). Training data are drawn from algorithmic programming settings, whereas evaluation includes out-of-distribution languages such as Go, PHP, C#, C, and JavaScript, together with Research and Production scenarios. The primary evaluation metric is macro-F1, which emphasizes balanced detection quality across the two labels instead of rewarding majority-class predictions.

2.2 Task Complexity and OOD Generalization

The democratization of Large Language Models (LLMs) has prioritized AI-generated code detection as a critical academic and security requirement (Becker et al., 2023; Ha et al., 2023; Perry et al., 2023). SemEval-2026 Task 13 formalizes this challenge within a landscape of *syntactic convergence*, where AI-generated code often imitates human-written idiomatic patterns (Guo et al., 2025; Orel et al., 2025b). Forensic boundaries are further obscured by high code similarity and clones between human-written and AI-generated repositories. Consequently, standard detectors often perform near randomly on programming code (Pan et al., 2024). A primary bottleneck remains multi-dimensional Out-of-Distribution (OOD) generalization (Pan et al., 2024). Robust systems must capture authorship markers that remain invariant across unseen languages and unseen functional domains, as emphasized in recent multi-lingual forensic benchmarks (Orel et al., 2025a,b).

2.3 Paradigms in AI-Generated Code Detection

Zero-shot and Statistical Detection. Zero-shot methods identify synthetic curvature and perplexity (DetectGPT (Mitchell et al., 2023), Binoculars (Hans et al., 2024; Nguyen and Akilesh, 2025)). Code-domain adaptations like Approximated Task Conditioning (ATC) (Ashkenazi et al., 2025) and DetectCodeGPT (Shi et al., 2025) address low-entropy masking. Regeneration approaches (DNA-GPT (Yang et al., 2023)) and fast-inference models (Bao et al., 2024; Fu et al., 2025) further optimize forensic scalability.

Neural Encoders and Supervised Learning. Foundational models like CodeBERT (Feng et al., 2020) and GraphCodeBERT (Guo et al., 2021) provide the basis for code understanding, yet primarily cluster by programming language (Pan et al., 2024; Pham et al., 2024). Advanced supervised frameworks like StyleDecipher (Li et al., 2025) and MAGECODE (Pham et al., 2024) integrate stylistic features with semantic embeddings. Contrastive frameworks (ConDA (Ma et al., 2024), UniCoR (Yang et al., 2025)) align features across domains to mitigate humanizing paraphrasing attacks (Guo et al., 2025).

PEFT and Agentic Reasoning. Parameter-Efficient Fine-Tuning (PEFT) via QLoRA (Dettmers et al., 2023) on models like Qwen-Coder (Hui et al., 2024) has standardized forensic specialization. Moving beyond monolithic models, Agentic Context Engineering (ACE) (Zhang et al., 2025) utilizes multi-agent adversarial debates to resolve classification uncertainty for ambiguous samples (Orel et al., 2025a,b).

2.4 Shift Toward Alignment-Driven Structural Artifacts

Traditional stylometry (Caliskan-Islam et al., 2015), focusing on individual habits, is increasingly diluted by LLM "consensus" styles. We hypothesize that discriminative signals have shifted toward *alignment-driven artifacts* (Achiam et al., 2023). Reinforcement Learning from Human Feedback (RLHF) optimizes for readability (Ouyang et al., 2022), inadvertently introducing systematic formatting regularities like standardized block separation and consistent whitespace. We formalize this as the Whitespace Alignment Artifact, quantified via the Empty-Line Ratio (R_{void}) (defined in §3.1). Vali-

dated empirically in §4.1, the integration of structural priors within our multi-view cascade improves reliability across OOD evaluation settings despite adversarial stripping risks (Pan et al., 2024).

3 Structure-Aware Contrastive Cascade

We propose the Structure-Aware Contrastive Cascade, a detection framework designed to address *syntactic convergence* by grounding generative likelihoods with explicit structural signals. To optimize throughput, the architecture uses a hierarchical scoring pipeline that escalates to agentic reasoning only when epistemic uncertainty is high. The system comprises four integrated modules, each serving a distinct purpose in the overall decision process.

3.1 Phase 1: Multi-View Stylometric Extraction

Phase 1 converts raw source code into an explicit forensic summary. It extracts a feature vector $\mathbf{f}(x)$ across three axes: structural bias via the Empty Line Ratio ($R_{void} = \frac{\#\text{blank lines}}{\#\text{total lines}}$) and indentation consistency; lexical predictability using character and bigram Shannon entropy ($H = -\sum p_i \log p_i$); and syntactic variance in naming conventions and line lengths. The purpose of this phase is to expose presentation-level cues that may be missed by a purely semantic model. These metrics are serialized into a short "Forensic Report" and injected into the model context.

3.2 Phase 2: Feature-Aware Fine-Tuning

Phase 2 adapts a code-optimized Large Language Model using Parameter-Efficient Fine-Tuning (PEFT) (Dettmers et al., 2023). The model is trained via next-token prediction on instruction prompts where $\mathbf{f}(x)$ is prepended to the code snippet. The purpose of this phase is to teach the model how to interpret structural evidence jointly with the snippet itself instead of treating the two signals independently. To improve robustness, we apply stochastic augmentation during training, including identifier renaming, whitespace perturbation, and comment stripping, guided by predefined probability thresholds.

3.3 Phase 3: Contrastive Inference

Phase 3 performs the main classification step through a contrastive label-loss mechanism. For snippet x, we measure the model's relative "surprise" via token-averaged Negative Log-Likelihood (NLL) for Human ($\mathcal{T}_H$) and AI ($\mathcal{T}_A$) hypotheses:

$$\mathcal{L}(y \mid x, \mathcal{T}) = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{<t}, x, \mathcal{T}) \tag{1}$$

The raw signal $\Delta_\ell = \mathcal{L}(\text{Human} \mid x, \mathcal{T}_H) - \mathcal{L}(\text{AI} \mid x, \mathcal{T}_A)$ is fused into the final ensemble score:

$$S_{final} = \lambda \cdot \sigma(\Delta_\ell) + (1 - \lambda) \cdot \hat{R}_{void} \tag{2}$$

where $\hat{R}_{void} = \min(2 \cdot R_{void}, 1.0)$ and λ is a tunable fusion weight. The purpose of this phase is to combine learned generative evidence with a lightweight structural prior in a single score. The decision $\hat{y} = \mathbb{I}(S_{final} > \tau)$ uses a threshold τ optimized via the Youden Index.

3.4 Phase 4: Uncertainty Resolution via Agentic Debate

Phase 4 is a selective refinement stage for borderline cases. Samples falling within a calibrated uncertainty margin ϵ trigger a multi-agent debate step. Three personas—Advocate (AI), Skeptic (Human), and Judge—evaluate the snippet against a forensic context playbook. The purpose of this phase is not to replace the cascade, but to revisit only those examples where Phase 3 remains uncertain. The agents operate at distinct temperature settings to balance diversity with deterministic judgment, producing a consensus verdict used to refine the final prediction.

4 Experimental Setup

4.1 Pipeline

Figure 1 illustrates the data flow and processing nodes of our framework. The pipeline sequentially maps to the phases described in Section 3: raw source code is first processed by the extraction node (Phase 1) to generate the forensic report. This data flows into the fine-tuned LLM node (Phase 2) and the contrastive scoring node (Phase 3) to compute generative likelihoods and structural metrics. Finally, samples exceeding the decision boundary uncertainty margin are routed to the multi-agent reasoning node (Phase 4) for consensus evaluation.

4.2 Dataset and Preprocessing

We utilize the official SemEval-2026 Task 13 corpus, which contains a near-balanced distribution of 238K human-written and 262K AI-generated samples from the 500K algorithmic training corpus (Orel et al., 2026a). Due to computational

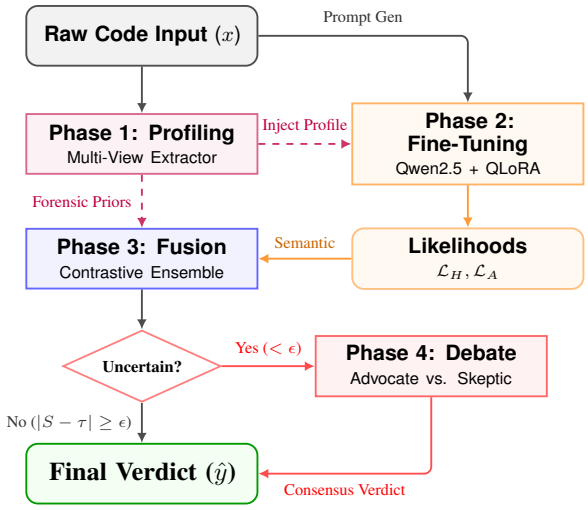

Figure 1: The Structure-Aware Contrastive Cascade pipeline

constraints, we sampled a curated 270K subset for training and reserved 30K for validation, strictly enforcing a **project-level split** to prevent stylistic leakage across partitions.

Feature Validation. To verify the *Whitespace Alignment Artifact*, we conducted a statistical analysis on the training set. A Welch's t-test confirmed a significant divergence in R_{void} between AI-generated and human-written code ($t = 47.2, p < 0.001$, Cohen's $d = 0.32$), validating the use of structural density as a language-agnostic forensic marker.

Stochastic Augmentation. As defined in §3.2, we implement a stochastic augmentation pipeline (Figure 2) to ensure robustness across unseen domains: **Identifier Renaming** ($p = 0.4$), **Whitespace Perturbation** ($p = 0.3$) within a $\pm 10\%$ density margin, and **Comment Stripping** ($p = 0.2$) (Orel et al., 2025b).

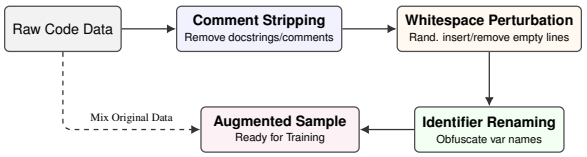

Figure 2: The Data Augmentation Pipeline. Training data undergoes stochastic transformation including comment stripping, whitespace perturbation, and identifier renaming to force the model to learn structural invariants.

4.3 Implementation Details

We fine-tuned **Qwen2.5-Coder-14B-Instruct** on a single NVIDIA H100 80GB GPU. Training utilized the AdamW optimizer with a learning rate of 1×10^{-5} and a cosine decay schedule over 2,500 steps (effective batch size 64).

PEFT Configuration. We applied QLoRA ($r = 64, \alpha_{lora} = 64$) (Dettmers et al., 2023) targeting all linear projection modules. Models were quantized to 4-bit NormalFloat (NF4) with double quantization to balance forensic reasoning capacity with memory efficiency. Full hyperparameter configurations are provided in **Appendix A.1**.

4.4 Inference and Calibration

The inference engine executes the hybrid fusion defined in Eq. (2), setting the semantic weight to $\lambda = 0.82$ based on validation set tuning.

Threshold Optimization. Decision thresholds (τ) were calibrated independently for each seen language using the Youden Index to maximize diagnostic power (Nguyen and Akilesh, 2025). For OOD languages and domains, we applied a **Global Default Threshold** ($\tau_{global} = 0.54$) derived from the pooled validation data to strictly avoid test-time leakage.

Agentic Refinement. The Multi-Agent Adversarial Debate (Phase 4) is triggered only when the system falls within the uncertainty margin of $\epsilon = 0.08$ ($|S_{final} - \tau| < 0.08$). The "Advocate" and "Skeptic" agents (frozen backbone, $T = 0.7$) perform one round of reasoning to resolve ambiguous cases near the decision boundary (Zhang et al., 2025). A deterministic "Judge" ($T = 0.0$) renders the final verdict based on the forensic rubrics in our Context Playbook.

4.5 Baselines

We benchmark our **Structure-Aware Contrastive Cascade** system against three established paradigms:

1. **Supervised PLMs:** CodeBERT (Feng et al., 2020), StarCoder2, and CodeT5+ (Pan et al., 2024) fine-tuned with a binary classification head.

2. **Zero-Shot Metrics:** DetectGPT (Mitchell et al., 2023) and Binoculars (Hans et al., 2024), utilizing Qwen-2.5-Coder-7B as the base observer model.

3. **Traditional Stylometry:** An XGBoost classifier trained on 23 handcrafted features including keyword density and cyclomatic complexity (Caliskan-Islam et al., 2015).

5 Results and Analysis

5.1 Official Leaderboard and Baselines

Our Structure-Aware Contrastive Cascade achieved 3rd place among 82 participating teams on the official SemEval-2026 Task 13A leaderboard. Table 1 presents the Top 10 rankings alongside established baselines.

Rank	Team / Method	Macro F1
1	TeleAI	0.997
2	Alejandro Mosquera	0.824
3	UIT-AMMC (Ours)	0.802
4	CEIA	0.753
5	Juan David Villate	0.752
6	Jany-Gabriel Ispas	0.737
7	Ruslan Berdichevsky	0.730
8	UIT_DoubleDat	0.730
9	QQQ	0.722
10	mcdok	0.697

Table 1: Official Task 13A leaderboard (Top 10) and baseline comparison. Our system substantially outperforms zero-shot and neural-stylometric methods.

5.2 Component Ablation

Table 2 isolates the contribution of each module. The structural prior (R_{void}) alone achieves F1=0.679, confirming the value of alignment-driven artifacts. The fusion-based cascade (S_{final} with $\lambda = 0.82$) yields 0.785, while the debate-based refinement step increases the final score to 0.802.

Configuration	Test F1	Δ
Pure R_{void} (Thresholding)	0.679	–
SFT (Qwen-Coder-14B)	0.740	+0.061
SFT + Statistical Features in Prompt	0.785	+0.106
Full Cascade (+ Agentic Debate)	0.802	+0.123

Table 2: Ablation study on the official test split. Agentic reasoning effectively corrects borderline errors where generative likelihoods are ambiguous.

5.3 Artifact Generalization

We validated the R_{void} artifact on the 1,000-sample diagnostic subset (Table 3). The effect size on the test set (Cohen's $d = 0.97$) is substantially larger than on the training corpus ($d = 0.32$), confirming that alignment-driven structural signals remain invariant and even amplify when encountering OOD generators (Guo et al., 2025).

Split	Human μ	AI μ	Cohen's d	p-value
Train	11.7%	15.4%	0.32	<0.001
Test	7.25%	16.50%	0.97	<0.001

Table 3: R_{void} artifact across data splits. Statistically significant divergence confirmed via Welch's t-test ($t = 12.03$).

5.4 Per-Language and OOD Analysis

Table 4 evaluates the system across all eight diagnostic languages, including the five OOD syntaxes defined in the task. C# and Go achieve high performance because their stricter formatting conventions make layout regularities more stable across repositories. By contrast, PHP remains challenging due to signal inversion ($d < 0$) caused by mixed-syntax templates (Guo et al., 2025).

Lang.	N	F1 (R_{void})	F1 (Cascade)	OOD
C#	122	0.810	0.842	✓
C++	75	0.744	0.782	
Go	60	0.744	0.781	✓
C	51	0.735	0.773	✓
JS	85	0.724	0.762	✓
Python	303	0.708	0.795	
Java	256	0.607	0.744	
PHP	48	0.238	0.452	✓

Table 4: Per-language performance on the diagnostic subset. The cascade improves detection across all seen and OOD languages, but the gains are uneven.

Decision thresholds τ were optimized for seen languages using the Youden Index: Python 0.53, Java 0.56, C++ 0.54, and an OOD global default of 0.54. The strongest languages in our setting are those where stylistic conventions remain comparatively rigid. C# and Go benefit from regular block formatting, explicit delimiters, and less template mixing, so the empty-line and indentation signals remain aligned with authorship cues. Java and Python improve under the full cascade, but they still show wider stylistic variability because educational repositories, framework boilerplate, and differing formatter choices blur structural boundaries.

PHP is the clearest failure case (F1=0.452). Many PHP files interleave executable code with

HTML templates, short tags, or view-layer boilerplate, so R_{void} often reflects page-layout structure rather than code-only formatting conventions. This leads to a negative effect size for the structural feature, meaning the direction of the signal can invert relative to the other languages. In addition, PHP snippets in the diagnostic subset are fewer in number and more heterogeneous in genre, which makes calibration harder. These observations suggest that future PHP-specific handling should normalize mixed markup-code regions or compute structural statistics after template-aware segmentation.

5.5 Forensic Error Analysis

A manual audit of 100 misclassified samples, reviewed by a single annotator, suggested that false positives primarily stem from educational code (45%) and boilerplate (30%), while false negatives often arise from compact AI generations (40%). We therefore treat these proportions as descriptive rather than definitive. Likewise, the observation that the debate stage corrected 14% of the audited errors should be interpreted cautiously: it is based on a small, single-annotator sample and does not constitute a controlled precision estimate for the debate module. A stronger evaluation would require multi-annotator agreement and a larger quantitative comparison between debated and non-debated cases (Zhang et al., 2025).

5.6 Statistical Reporting

All results use macro-averaging across classes (diagnostic subset: 77.7% human-written, 22.3% AI-generated). Bootstrap 95% CIs (1000 resamples, seed=42): Full Cascade F1=0.802 [0.788, 0.816] computed on the official test split; R_{void} F1=0.679 [0.647, 0.710] on the diagnostic subset.

6 Conclusion

In this study, we addressed the challenge of binary AI-generated code detection by exploring the limitations of existing semantic-heavy paradigms. Our experiments confirmed that traditional XGBoost stylometry, zero-shot perplexity methods, and monolithic contrastive models fail to maintain robustness under the pressure of *syntactic convergence*. By identifying the Whitespace Alignment Artifact as a language-agnostic forensic signal, we developed the Structure-Aware Contrastive Cascade to bridge structural heuristics and large-scale generative reasoning.

Our approach achieved a Macro F1 score of 0.802, placing 3rd among 82 participating teams in the SemEval-2026 Task 13 competition. The $+0.123$ improvement over pure structural thresholding supports the synergy between forensic priors and model-based inference.

Limitations. Despite its efficacy, our structural marker remains sensitive to mixed-syntax environments like PHP (F1=0.452) and is potentially vulnerable to adversarial formatting perturbations. In addition, the manual error analysis was based on a single annotator and a small audited subset, so conclusions about the debate step should be interpreted cautiously.

Future Work. Future efforts will focus on enhancing the agentic refinement mechanism (Zhang et al., 2025). We also aim to investigate AST-level structural signals and develop robust defense mechanisms against formatting-based evasion attacks.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Jibi Ajiam, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Maor Ashkenazi, Shachar Langbeheim, Shai Shalev-Shwartz, and Amit Daniely. 2025. Zero-shot detection of llm-generated code via approximated task conditioning. *arXiv preprint arXiv:2506.06069*. Accepted at ECML-PKDD 2025.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *Proceedings of ICLR*.

Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 500–506, New York, NY, USA. Association for Computing Machinery.

Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss Joint, Fabian Yamaguchi, and Rachel Greenstadt. 2015. De-anonymizing programmers via code stylometry. In *Proceedings of the 24th USENIX Security Symposium*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Proceedings of NeurIPS*.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. Codebert: A pre-trained model for programming and natural languages. In *Proceedings of EMNLP*.

Jiachen Fu, Siyuan Li, and 1 others. 2025. Detectanyllm: Towards generalizable and robust detection of machine-generated text. In *Proceedings of ACM Multimedia (MM '25)*.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, and 1 others. 2021. Graphcodebert: Pre-training code representations with data flow. In *Proceedings of ICLR*.

Hanxi Guo, Siyuan Cheng, Kaiyuan Zhang, Bohong Wu, Siyuan Chen, and Zhidong Deng. 2025. Codemirage: A multi-lingual benchmark for detecting ai-generated and paraphrased source code. *arXiv preprint arXiv:2506.11059*.

Huyen Ha, Duc Tran, and Dukyun Kim. 2023. Black-box adversarial attacks against language model detector. In *Proceedings of the 12th International Symposium on Information and Communication Technology (SoICT 2023)*.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamed Fendley, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text. In *Proceedings of ICML*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Siyuan Li, Jiachen Fu, Di Fu, Cheng-Chieh Huang, and 1 others. 2025. Styledecipher: Robust and explainable detection of llm-generated texts with stylistic analysis. *arXiv preprint arXiv:2510.12608*.

Zeyu Ma, Hongzhan Chen, Hongbo Xu, Jianzheng Liu, Lingyong Yan, Wei He, and Xueqi Cheng. 2024. Conda: Contrastive domain adaptation for ai-generated text detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6524–6538.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Trieu Hai Nguyen and Sivaswamy Akilesh. 2025. Vietbinoculars: A zero-shot approach for detecting vietnamese llm-generated text. *arXiv preprint arXiv:2509.26189*.

Daniil Orel, Dilshod Azizov, and Preslav Nakov. 2025a. Codet-m4: Detecting machine-generated code in multi-lingual settings. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov. 2026a. AICD bench: A challenging benchmark for ai-generated code detection. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, Rabat, Morocco. Association for Computational Linguistics. (Accepted).

Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov. 2026b. SemEval-2026 task 13: Detecting machine-generated code with multiple programming languages, generators, and application scenarios. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*, San Diego, USA. Association for Computational Linguistics.

Daniil Orel, Indraneil Paul, Iryna Gurevych, and Preslav Nakov. 2025b. `Droid`: A resource suite for AI-generated code detection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31263–31289, Suzhou, China. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Proceedings of NeurIPS*.

Wei Hung Pan, Cheng-Chieh Huang, Yan-Ling Chang, and Huan-Chao Keh. 2024. Assessing ai detectors in identifying ai-generated code: Implications for education. In *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 321–327. ACM.

Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do users write more insecure code with ai assistants? In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*.

Hung Pham, Huyen Ha, Van Tong, Duc Tran, and Dukyun Kim. 2024. Magecode: Machine-generated code detection method using large language models. *IEEE Access*, 12:164230–164245.

Yuling Shi, Hongyu Zhang, Chengcheng Wan, and Xiaodong Gu. 2025. *Between Lines of Code: Unraveling the Distinct Patterns of Machine and Human Programmers*, page 1628–1639. IEEE Press.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Petzold, William Yang Wang, and Haifeng Chen. 2023. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. In *Proceedings of EMNLP*.

Yang Yang, Xinyu Li, Duyu Tang, and 1 others. 2025. Unicor: Modality collaboration for robust cross-language hybrid code retrieval. In *Proceedings of ICSE 2026*.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Shubham Toshniwal, Yashar Mehdad, Veselin Stoyanov, and Sewon Min. 2025. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*. Accepted at ICLR 2026.

A Appendix

A.1 Hyperparameter Configuration

Table 5 details the high-capacity LoRA configuration used for fine-tuning.

Hyperparameter	**Value**
Quantization	4-bit NF4 (Double Quant)
LoRA Rank (r) / Alpha (α)	64 / 64
Target Modules	All Linear Layers
Batch Size	4 (Grad Accum = 16)
Learning Rate	1×10^{-5} (Cosine Decay)
Max Sequence Length	2,048

Table 5: Training hyperparameters. The high rank ($r = 64$) is critical for authorship tasks.

Algorithm 1 Robust Instruction Tuning with Statistical Feature Injection

Require: Dataset $\mathcal{D} = \{(c_i, l_i, y_i)\}_{i=1}^{N}$ where c is code, l is language, and $y \in \{0, 1\}$ is label (0: Human, 1: AI)
Require: Pretrained LLM $\mathcal{M}_\Theta$, LoRA parameters Φ
Require: Augmentor $\mathcal{A}$, Statistics Extractor $\mathcal{S}$, Prompt Template $\mathcal{T}$
Initialize LoRA adapters Φ (e.g., rank $r = 64$, $\alpha = 64$)
for each epoch **do**
  for each minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
    $\mathcal{X}_{\text{batch}} \leftarrow \emptyset$
    for each $(c, l, y) \in \mathcal{B}$ **do**
      {Data augmentation: rename variables, perturb whitespace, strip comments}
      $c' \leftarrow \mathcal{A}(c)$
      {Feature extraction: empty_lines, entropy, length statistics}
      $s \leftarrow \mathcal{S}(c')$
      {Construct instruction prompt with statistical profile}
      $target \leftarrow$ `"AI"` if $y = 1$ else `"Human"`
      $x_{\text{input}} \leftarrow \mathcal{T}(c', l, s)$
      $x_{\text{full}} \leftarrow x_{\text{input}} \oplus target$
      $\mathcal{X}_{\text{batch}} \leftarrow \mathcal{X}_{\text{batch}} \cup \{x_{\text{full}}\}$
    end for
    {Optimize cross-entropy loss on next-token prediction}
    $\mathcal{L}_{\text{SFT}} \leftarrow -\frac{1}{|\mathcal{X}_{\text{batch}}|} \sum_{x \in \mathcal{X}_{\text{batch}}} \sum_{t=1}^{|x|} \log P(x_t \mid x_{<t}; \Theta, \Phi)$
    Update $\Phi \leftarrow \text{AdamW}(\nabla_\Phi \mathcal{L}_{\text{SFT}})$
  end for
end for
return Fine-tuned adapters Φ^*

Algorithm 2 Contrastive Inference with Structural Fusion

Require: Test code c, language l, fine-tuned model $\mathcal{M}_{\Theta,\Phi^*}$
Require: Statistics extractor $\mathcal{S}$, prompt template $\mathcal{T}$, language-specific threshold $\tau(l)$
Ensure: Predicted label $\hat{y} \in \{\text{Human}, \text{AI}\}$
{Compute structural prior}
$s \leftarrow \mathcal{S}(c)$ {includes R_{void}}
$R_{void} \leftarrow s$[`empty_line_ratio`]
$\hat{R}_{void} \leftarrow \min(2 \cdot R_{void}, 1.0)$
{Construct hypothesis prompts}
$h_{\text{H}} \leftarrow \mathcal{T}(c, l, s) \oplus$ `" Human"`
$h_{\text{A}} \leftarrow \mathcal{T}(c, l, s) \oplus$ `" AI"`
{Compute token-averaged NLL for each hypothesis}
$\mathcal{L}_{\text{H}} \leftarrow -\frac{1}{T_H} \sum_{t=1}^{T_H} \log P((h_{\text{H}})_t \mid (h_{\text{H}})_{<t}; \Theta, \Phi^*)$
$\mathcal{L}_{\text{A}} \leftarrow -\frac{1}{T_A} \sum_{t=1}^{T_A} \log P((h_{\text{A}})_t \mid (h_{\text{A}})_{<t}; \Theta, \Phi^*)$
$\Delta_\ell \leftarrow \mathcal{L}_{\text{H}} - \mathcal{L}_{\text{A}}$
$S_{\text{sem}} \leftarrow \sigma(\Delta_\ell)$
$S_{\text{final}} \leftarrow \lambda \cdot S_{\text{sem}} + (1 - \lambda) \cdot \hat{R}_{void}$ {e.g., $\lambda = 0.82$}
if $S_{\text{final}} > \tau(l)$ **then**
  $\hat{y} \leftarrow$ AI
else
  $\hat{y} \leftarrow$ Human
end if
return $\hat{y}, S_{\text{final}}$

Algorithm 3 ACE-inspired Multi-Agent Debate Refinement

Require: Code c, language l, base prediction $(\hat{y}, S_{\text{final}})$ from Alg. 2
Require: Debate model $\mathcal{M}_{\Theta,\Phi^*}$, uncertainty margin ϵ
Ensure: Refined label $\hat{y}_{\text{refined}}$
if $|S_{\text{final}} - \tau(l)| \geq \epsilon$ **then**
  return $\hat{y}$ {Confident case, no debate}
end if
{Initialize three personas: Advocate, Skeptic, Judge}
context $\leftarrow$ ConstructDebateContext$(c, l, S_{\text{final}}, \hat{y})$
$\text{arg}_{\text{AI}} \leftarrow$ Generate$(\mathcal{M}$, "Advocate", context, $T = 0.7)$
$\text{arg}_{\text{H}} \leftarrow$ Generate$(\mathcal{M}$, "Skeptic", context, $T = 0.7)$
$\hat{y}_{\text{refined}} \leftarrow$ JudgeDecision$(\mathcal{M}$, "Judge", context, $\text{arg}_{\text{AI}}, \text{arg}_{\text{H}}, T = 0.0)$
return $\hat{y}_{\text{refined}}$