Bitzkrieg at SemEval-2026 Task 13: Calibration-Aware Dual CodeBERT for Multilingual Machine-Generated Code Detection

Thenmozhi D **Adithya S** **Harshil Malisetty** **Aadit P** **Rohan R***
Department of Computer Science and Engineering
Sri Sivasubramaniya Nadar College of Engineering
Chennai, India
{theni_d, adithya2410422, harshil2410093}@ssn.edu.in
{aadit2410180, rohan2210124}@ssn.edu.in

Abstract

We describe our submission to SemEval-2026 Task 13 (Orel et al., 2026), addressing binary detection (Subtask A), generator attribution (Subtask B), and hybrid/adversarial authorship classification (Subtask C) of machine-generated code (MGC). For Subtask A, we fine-tune two CodeBERT (Feng et al., 2020) models with complementary sampling strategies and apply percentile-based post-hoc calibration, improving Macro-F1 from 0.47 to 0.56 without additional training. For Subtask B, we combine TF-IDF n-grams, frozen CodeBERT embeddings, and language features with XGBoost (Chen and Guestrin, 2016), using synthetic augmentation and class weighting to handle an 11-class dataset skewed 88% toward the human class, achieving Macro-F1 of 0.289. For Subtask C, we fine-tune a CodeBERT classifier for four-way authorship classification, achieving Macro-F1 of 0.49. Our results highlight the importance of probability calibration for binary detection and class balancing for multi-class attribution.

1 Introduction

Recent advances in LLMs have had an undeniable impact on software development. Generative models are now able to produce syntactically correct and semantically relevant code for a variety of programming languages, further blurring the distinction between human- and machine-written code (Devlin et al., 2019; Liu et al., 2019; Wolf et al., 2020). As a result, there has been growing difficulty in distinguishing between machine-generated code (MGC) and human-written code (HWC), which complicates validating academic integrity, code review, and securing the software supply chain. This growing reliance on automated code generation has motivated research into reliable detection and attribution methods for machine-generated code.

SemEval-2026 Task 13 (Orel et al., 2026) is a three-part task intended to provide solutions to these issues on a multi-domain and multilingual basis. Subtask A determines whether code is HWC or MGC. Subtask B identifies the specific generator: human or one of 10 different LLM families. Subtask C classifies code into four groups: human, machine, hybrid, and adversarial. Each subtask uses evaluation datasets composed of programming languages and domains not found in training.

2 Task Setting

2.1 Subtask A: Binary Detection

The training data for Subtask A consists of 500,000 examples across three programming languages: C++, Python, and Java, all from the competitive programming domain. The goal is to classify code as fully human-produced or fully machine-generated. Evaluation covers four test configurations: seen languages with seen completions; unseen languages (Go, PHP, C, C#, JavaScript) with seen completions; seen languages with unseen completions; and unseen languages with unseen completions. Performance is measured using Macro-F1 (Sokolova and Lapalme, 2009), which assigns equal weight to each class and is appropriate for imbalanced settings.

2.2 Subtask B: Multi-Class Attribution

Subtask B uses the same 500,000-example dataset. The classification target is 11-way: human (class 0) or one of 10 LLM families (classes 1–10): DeepSeek-AI, Qwen, 01-ai, BigCode, Gemma, Phi, Meta-LLaMA, IBM-Granite, Mistral, and OpenAI. The training dataset is severely imbalanced: 442,096 examples are human (88%) and between 1,968 and 10,810 examples per LLM class (0.4%–2%).

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 2233–2237

2.3 Subtask C: Hybrid and Adversarial Authorship Classification

Subtask C is a four-way classification task capturing fine-grained authorship categories: human (0), fully machine-generated (1), hybrid (2, partially completed or rewritten by LLMs), and adversarial (3, purposely generated to imitate human style). Unlike binary detection, this task reflects real-world AI-assisted development where authorship exists on a spectrum. Hybrid and adversarial samples create ambiguity by sharing features with both HWC and MGC, complicating separation. The main metric is Macro-F1.

3 Preliminary Experiments

Preliminary trials tested baseline feature-based methods, XGBoost pipelines, and CodeBERT fine-tuning. For Subtask A, heuristic feature-based methods achieved a maximum Macro-F1 of 0.44, while CodeBERT fine-tuning improved this to 0.47 but proved unstable across languages and domains. For Subtask B, a TF-IDF-based XGBoost showed strong bias towards the dominant human class, achieving Macro-F1 of 0.28. For Subtask C, baseline CodeBERT models produced 0.45–0.48 Macro-F1, with high confusion between hybrid and adversarial samples.

4 Method

4.1 Subtask A

4.1.1 Backbone Architecture

All Subtask A models use `microsoft/codebert-base` (Feng et al., 2020), built on the BERT architecture (Devlin et al., 2019). Each classifier consists of a CodeBERT encoder, a dropout layer (rate 0.2), and a linear classification head mapping the 768-dimensional `[CLS]` representation to two output logits:

$$h = \text{CodeBERT}(x)_{[\text{CLS}]} \tag{1}$$

$$z = Wh + b \tag{2}$$

$$p = \text{softmax}(z) \tag{3}$$

4.1.2 Two-Brain Training Strategy

To address imbalanced language representation, two complementary models were trained using different data sampling strategies with the HuggingFace Transformers library (Wolf et al., 2020).

Brain A (Balanced Polyglot). Capped at 8,000 samples per language, forcing the model to learn language-invariant signal rather than exploiting majority-language statistics. Training configuration: learning rate 2×10^{-5}, batch size 16, 2 epochs, max sequence length 256, AdamW optimizer.

Brain B (Specialist). Trained on the full unbalanced dataset for 1 epoch with the same hyperparameters, allowing sharper decision boundaries for dominant languages.

No external data or third-party AI detectors were used.

4.1.3 Percentile-Based Calibration

Raw ensemble probabilities are poorly calibrated across languages. Instead of the standard 0.5 decision boundary, we apply post-hoc percentile thresholding: the threshold is set to the α-th percentile of validation scores. We sweep $\alpha \in \{0.10, 0.25, 0.30\}$ and find that $\alpha = 0.25$ and $\alpha = 0.10$ both plateau at Macro-F1 0.56, indicating strong ranking ability but limited score separability.

4.2 Subtask B

4.2.1 Feature Engineering

For the 11-class attribution problem, we combine three complementary feature types (see Figure 1):

TF-IDF Features. Character n-grams (3–5, max 8,000 features) and word n-grams (1–2, max 5,000 features) with minimum document frequency 3 (Salton and Buckley, 1988). Character n-grams capture low-level stylistic patterns; word n-grams encode higher-level structural choices.

CodeBERT Embeddings. Pre-extracted 768-dimensional `[CLS]` embeddings from `microsoft/codebert-base`. Due to computational constraints, frozen embeddings were used rather than fine-tuning end-to-end for 11 classes. These provide semantic representations complementary to surface n-gram features.

Language Features. One-hot encoded programming language (8 languages), providing direct language-specific hints since LLM code styles vary by language.

4.2.2 Training Strategy

To counter the extreme class imbalance (88% human samples), we use synthetic augmentation and class weighting.

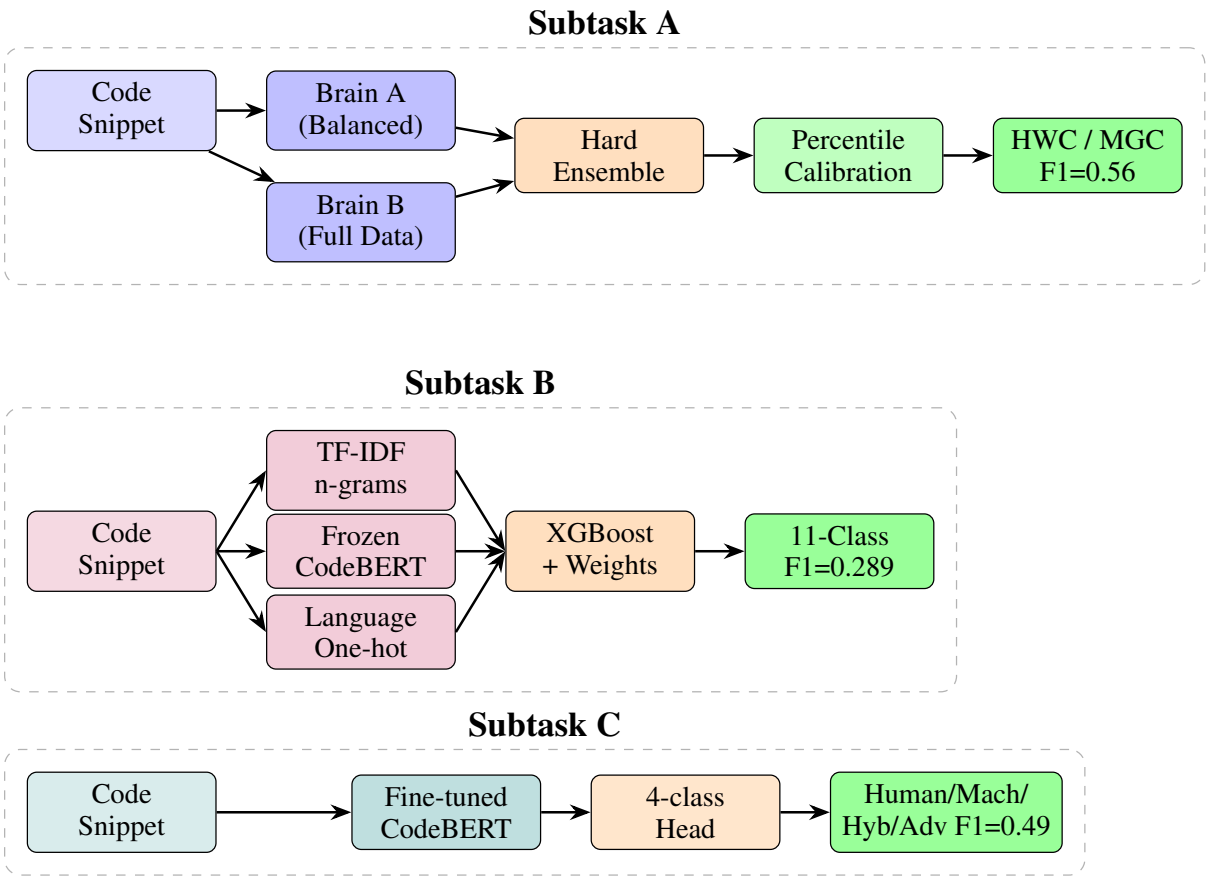

Figure 1: System pipelines for all three subtasks. Subtask A uses dual CodeBERT models with post-hoc percentile calibration. Subtask B fuses TF-IDF n-grams, frozen CodeBERT embeddings, and language features into XGBoost. Subtask C fine-tunes a single CodeBERT with a four-class head.

Synthetic Augmentation. Each rare LLM class with fewer than 10,000 samples is oversampled with replacement to 10,000 samples, adding minor perturbations such as appending comment markers. This yields a balanced training set of 542,906 samples.

XGBoost with Class Weights. We train XGBoost (Chen and Guestrin, 2016) (200 trees, max depth 7, learning rate 0.1) with class weights raised to power 1.2 to further emphasize rare classes. XGBoost handles the heterogeneous 13,776-dimensional feature space effectively (8,000 char + 5,000 word + 8 language + 768 CodeBERT), and its gradient-boosting objective is well-suited to the imbalanced multi-class setting.

4.3 Subtask C: CodeBERT-Based Authorship Classifier

For Subtask C, we fine-tune `microsoft/codebert-base` (Feng et al., 2020) for four-class classification. The architecture mirrors Subtask A: a CodeBERT encoder, dropout (0.1), and a linear head mapping the 768-dimensional `[CLS]` embedding to four output logits.

Training uses cross-entropy loss (LR 2×10^{-5}, batch size 16, max length 256) with dynamic padding and mixed precision. No external features were used, allowing the model to learn stylistic patterns directly from token sequences.

5 Results

5.1 Subtask A: Ablation Study

Table 1 shows an ablation over each component of the Subtask A pipeline.

Individual Brain A and Brain B models underperform a single fine-tuned CodeBERT on their own; their hard ensemble recovers to 0.47. Post-hoc calibration then adds a further 0.09 Macro-F1 improvement with no additional training.

Threshold Sweep. Evaluated multiple percentile values: $\alpha = 0.30 \rightarrow 0.53$, $\alpha = 0.25 \rightarrow 0.56$, $\alpha = 0.10 \rightarrow 0.56$. The plateau at $\alpha = 0.25$ and $\alpha = 0.10$ indicates strong ranking but limited

Model / Configuration	Macro-F1
Feature-based heuristic	0.44
Single CodeBERT (fine-tuned)	0.47
Brain A only (balanced)	0.40
Brain B only (full data)	0.39
Hard Ensemble (A + B)	0.47
+ Percentile Calibration (α=0.25)	**0.56**

Table 1: Subtask A ablation: each row adds one component. Calibration yields the largest single gain (+0.09 F1).

score separability.

Error Analysis. Common failure cases include short snippets lacking stylistic signals, boilerplate algorithm templates, and highly generic utility functions. These findings suggest the model relies on stylistic cues rather than deeper semantic reasoning.

5.2 Subtask B: Multi-Class Attribution Results

The final Subtask B system achieved Macro-F1 of 0.289 on the validation set. Table 2 shows per-class performance.

Class	F1	Support
Human (0)	0.90	88,490
DeepSeek-AI (1)	0.15	847
Qwen (2)	0.19	1,755
01-ai (3)	0.20	650
BigCode (4)	0.28	445
Gemma (5)	0.48	372
Phi (6)	0.47	1,118
Meta-LLaMA (7)	0.16	1,695
IBM-Granite (8)	0.43	1,579
Mistral (9)	0.14	895
OpenAI (10)	0.44	2,154
Macro Avg	**0.289**	**100,000**

Table 2: Subtask B per-class results (validation set).

The human class achieved high F1 (0.90) due to its abundance in training data. Mid-sized LLM classes (Gemma, Phi, IBM-Granite, OpenAI) achieved moderate F1 (0.43–0.48), while rare classes (DeepSeek-AI, Qwen, Meta-LLaMA, Mistral) struggled (0.14–0.20). The F1 gap between classes tracks closely with training support, indicating data sparsity rather than representational limitations as the primary bottleneck.

5.3 Subtask C: Hybrid and Adversarial Classification Results

The final Subtask C system achieved a Macro-F1 score of 0.49 on the official evaluation set. Table 3 shows per-class performance.

Class	F1 Score
Human	0.62
Machine	0.55
Hybrid	0.41
Adversarial	0.38
Macro Avg	**0.49**

Table 3: Per-class F1 scores for Subtask C (validation set).

Performance was highest for fully human and fully machine-generated samples, where stylistic cues are more distinct. Hybrid and adversarial samples proved more challenging due to overlapping characteristics and deliberate stylistic mimicry.

6 Discussion

Each subtask turned out to have a different bottleneck, which shaped how we approached them.

Subtask A: Calibration Matters. Binary classifiers may lack well-calibrated probabilities in multilingual settings. The dual-CodeBERT model demonstrated strong ranking ability but required calibrated thresholds to translate that into competitive Macro-F1. Applying percentile-based thresholds raised performance from 0.47 to 0.56 without any retraining, confirming that post-hoc calibration is an effective and low-cost improvement when the underlying classifier ranks well but has poorly-calibrated scores.

Subtask B: Class Imbalance Dominates. For multi-class attribution, the main challenge was the extreme class imbalance (88% human, 0.4–2% per LLM class) combined with distribution shift. Despite synthetic augmentation and class weighting, rare LLM classes (DeepSeek-AI, Qwen, Meta-LLaMA, Mistral) scored only 0.14–0.20 F1, while more frequent classes (Gemma, Phi, IBM-Granite, OpenAI) reached 0.43–0.48. The near-linear relationship between per-class support and per-class F1 points to data sparsity as the core limiting factor.

Subtask C: Authorship Spectrum Challenges. Hybrid and adversarial samples are hard to separate because they occupy the boundary between human and machine code. Hybrid samples mix human-authored and LLM-completed code; adversarial samples are machine-generated but stylistically manipulated to resemble human writing. Both produce overlapping feature distribu-

tions. Future work using structure-aware representations such as AST features and contrastive training objectives could better capture fine-grained authorial characteristics.

Cross-Task Insights. Across all three subtasks, task-specific post-processing contributed more than architectural choices. Calibration fixes binary detection, aggressive class balancing helps attribution, and richer structural features are what spectrum classification actually needs.

7 Conclusion

We tackled three related but distinct tasks: binary MGC detection (Subtask A), multi-class generator attribution (Subtask B), and spectrum classification across human, hybrid, adversarial, and machine-generated code (Subtask C).

For Subtask A, a dual-CodeBERT model with two independently fine-tuned branches and percentile-based calibration improved Macro-F1 from 0.40 to 0.56 without additional training, demonstrating the value of post-hoc calibration in multilingual settings.

For Subtask B, a pipeline combining TF-IDF n-grams, frozen CodeBERT embeddings, and language features with XGBoost and class weighting achieved Macro-F1 of 0.289. Performance was primarily limited by data sparsity in rare LLM classes rather than representational limitations.

For Subtask C, a fine-tuned CodeBERT classifier achieved Macro-F1 of 0.49. Accurate authorship detection across the spectrum requires task-specific methods: calibration-aware inference for binary detection and appropriate class balancing for multi-class attribution.

Limitations

There are several limitations across the three subtasks. In Subtask A, only one model branch was trained for a full epoch due to compute constraints, and calibrating thresholds using validation percentiles risks overfitting to the public leaderboard. In Subtask B, programming language was detected heuristically rather than using ground-truth labels, which introduces noise. In Subtask C, the architecture uses only token-based features, limiting its ability to distinguish hybrid from adversarial examples; the 256-token sequence length also truncates longer snippets. Future improvements include longer training, structure-aware representations (AST features), improved language detection, and better calibration for imbalanced multi-class data.

References

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

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

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, and Ting Liu. 2020. Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov. 2026. 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.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics.