Binary Machine-Generated Code Detection: A Comparative Study of CodeBERT, GraphCodeBERT, and Logit Averaging

Abdulla Al Messabi[†][*]
Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
`Abdulla.Almessabi@mbzuai.ac.ae`

Abstract

This paper presents a comparative study for SemEval-2026 Task 13 Subtask A, which proposes the binary classification of code snippets as either human-written or AI-generated across multiple programming languages and domains. In this work, we examine three modeling paradigms: (1) a TF–IDF enhanced logistic regression baseline, (2) fine-tuning of CodeBERT-base, and (3) fine-tuning of GraphCodeBERT. Considering the limited improvement over the baseline made by the individual transformers, we experimented with a simple logit-averaging ensemble of CodeBERT and GraphCodeBERT. This ensemble yields our best macro-F1 score of **0.3732** on the official test set. Our findings highlight that lightweight ensembling of structurally distinct transformer encoders can offer meaningful gains even when individual models plateau.

1 Introduction

With the rise of AI-powered coding assistants such as GitHub Copilot and ChatGPT, machine-generated code is increasingly common in software development workflows. This introduces practical challenges in authorship attribution and academic integrity. SemEval-2026 Task 13 Subtask A focuses on the detection of machine-generated code (Orel et al., 2025) through binary classification of code snippets.

The task is challenging due to:

- multilingual code (seen and unseen languages), produced by diverse generation models,
- multiple domains (algorithmic, research, production),
- large dataset size (500K+ samples).

[*]Corresponding author.

2 Methodology

2.1 Dataset

We used the official dataset from the HuggingFace repository `DaniilOr/SemEval-2026-Task13`. The dataset includes 500K training samples (mixed human/AI), 100K validation samples, in multilingual, multi-domain test sets. Training languages are C++, Python, and Java in algorithmic domains.

2.2 Baseline: TF–IDF + Logistic Regression

We first implemented the enhanced baseline provided for the task. TF–IDF features were computed using:

- 1–4 gram word/character n-grams,
- maximum of 150,000 features,
- normalized, lowercased tokens.

A logistic regression classifier with balanced class weights was trained on the sparse features. This model captures lexical frequency patterns, syntactic tokens, and stylistic artifacts typical of human or AI-generated code.

2.3 Fine-Tuning CodeBERT

We fine-tuned **microsoft/codebert-base**, a bidirectional transformer pre-trained on paired natural language and code. For this model, we used a balanced subset of the training data containing 200,000 samples per class. A balanced validation split of 40,000 samples per class was used.
During initial experiments, we observed unusually high validation macro-F1 scores, which strongly suggested that the model was overfitting rather than true generalization patterns. To mitigate this, training was restricted to **2 epochs**.

CodeBERT achieved a macro-F1 of **0.312** on the official test set.

2.4 Fine-Tuning GraphCodeBERT

The model showed even stronger tendencies toward rapid overfitting, while validation performance plateaued early. This indicated that the model was fitting dataset-level artifacts extremely quickly. Combined with the higher computational cost, we trained GraphCodeBERT for only **1 epoch** on a slightly smaller but balanced subset containing **150000** samples per class. Validation used 30000 samples per class.

Despite the reduced training size and epoch count, GraphCodeBERT achieved a stronger macro-F1 of **0.354**.

2.5 Logit Averaging Ensemble

Although GraphCodeBERT outperformed CodeBERT, both models individually remained close to baseline performance. Ensembling is a well-established strategy for improving robustness and reducing prediction variance (Dietterich, 2000). Motivated by this, we computed a simple unweighted average of the model logits to exploit their complementary strengths:

Let:
ModelA = CodeBERT
ModelB = GraphCodeBERT

$$p_{\text{CB}} = \text{logits}_{\text{ModelA}} \quad \text{and} \quad p_{\text{GCB}} = \text{logits}_{\text{ModelB}}.$$

$$p_{\text{final}} = 0.5 \cdot p_{\text{CB}} + 0.5 \cdot p_{\text{GCB}}.$$

A decision threshold of 0.6 was applied:

$$\hat{y} = \begin{cases} 1 & \text{if } p_{\text{final}} \geq 0.6, \\ 0 & \text{otherwise.} \end{cases}$$

This ensemble yielded a macro-F1 of **0.3732** on the official test set.

3 Results and Discussion

Model	Macro-F1
TF–IDF Baseline	∼0.30
CodeBERT	0.312
GraphCodeBERT	0.354
Ensemble (Logit Averaging)	**0.3732**

Table 1: Comparison of model performance.

3.1 Discussion

The ensemble-based approach leverages the complementary strengths of the two transformer encoders. This approach allowed for strong syntactic and lexical modeling, while incorporating structural information through data-flow dependencies, enabling the model to detect deeper semantic and control-flow characteristics.

3.2 Limitations

- **Overfitting during fine-tuning:** Both CodeBERT and GraphCodeBERT exhibited rapid overfitting, a known issue in fine-tuning large pretrained transformers (Mosbach et al., 2021). This forced the use of fewer epochs and smaller training subsets, which may cap overall performance.

- **Reduced hyperparameter exploration:** Due to computational constraints, we were unable to conduct systematic hyperparameter search. The reported results may therefore reflect suboptimal configurations.

- **Limited training data:** The training data focused exclusively on algorithmic code across three languages, which may restrict generalization to unseen domains.

4 Conclusion

We presented a comparative evaluation of classical and transformer-based approaches for machine-generated code detection. While TF–IDF and individual fine-tuned transformers achieve limited performance, a simple ensemble via logit averaging produces meaningful gains, reaching a macro-F1 of **0.3732**. This suggests that lightweight ensembling of structurally diverse code models can be an effective strategy when individual models plateau.

References

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *ICLR*.

Dario Orel, Danil Azizov, and Preslav Nakov. 2025. Codet-m4: Detecting machine-generated code in multilingual, multi-generator and multi-domain settings. In *Findings of ACL*.