Enhanced Code Comment Relevance Classification Using Domain-Adaptive CodeBERT with Masked Language Modeling

L. Simiyon Vinscent Samuel[1], Sundareswaran Rajaguru[1], Vijayaraaghavan K. S.[1] and A. Vijayalakshmi[1]

[1]*Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai,India*

Abstract

We evaluate domain-adaptive pre-training of CodeBERT using masked language modeling (MLM) for binary code-comment relevance classification in the IRSE shared task. We compare CodeBERT+MLM against TF–IDF baselines and study the effect of synthetic and silver-standard augmentation. CodeBERT+MLM attains F_1=0.9193 on the original test split versus 0.7067 for the best TF–IDF baseline (logistic regression). Augmentation (combined data) produces a marginal change (F_1=0.9166), highlighting that augmentation quality — not quantity — drives gains. We provide implementation details and practical recommendations for applying domain-adaptive pre-training in software engineering tasks.

Keywords

code comment classification, CodeBERT, domain-adaptive pre-training, masked language modeling, data augmentation

1. Introduction

Code comments are essential documentation artifacts that improve comprehension and maintenance [1, 2]. The IRSE shared task frames comment relevance as a binary classification problem: whether a comment meaningfully describes a code segment. While classical TF–IDF models often perform strongly on technical text [6], pre-trained transformers tailored to code (e.g., CodeBERT [5]) can capture richer cross-modal patterns. Domain-adaptive pre-training (DAPT) via continued MLM on domain text has been shown to close distributional gaps between generic pretraining and task data [7].

This paper investigates CodeBERT with intermediate MLM on C code-comment corpora and compares it to TF–IDF baselines, assessing the impact of rule-based synthetic and silver-standard augmentations. Our main contributions: (i) an empirical evaluation of CodeBERT+MLM on original and augmented data; (ii) comparisons with classical baselines; and (iii) analysis of augmentation effects and cross-dataset generalization.

2. Related work (condensed)

Transformers adapted for code, such as CodeBERT, enable joint code–text representations useful for search and summarization [5, 8]. DAPT/TAPT improves downstream task fit by continued MLM on domain/task data [7, 11]. Prior IRSE work shows TF–IDF models can be competitive for comment relevance, motivating careful empirical comparison [6]. Data augmentation (rule-based or model-based) helps low-resource settings but can introduce label noise in domain-specific tasks lacking automatic verification [12, 13]. [20, 21, 22] have similar ideas for the task.

Forum for Information Retrieval Evaluation, December 17-20, 2025, India

vijayaraaghavan.s.123@gmail.com (V. K. S.)

© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-02-24

CEUR-WS.org/Vol-4173/T6-3.pdf

3. Datasets and experimental setup

3.1. Corpora and silver-standard creation

We use the IRSE/FIRE seed dataset (11,452 code–comment pairs extracted from C projects). After cleaning malformed or unlabeled entries we obtain 8,326 training samples ("Original"). To expand data we used two augmentation sources:

1. **Rule-based synthetic:** 500 examples generated by templated paraphrasing (synonym substitution, sentence restructuring) derived from existing comments to increase stylistic diversity while preserving labels.
2. **HumanEval-derived silver standard:** 1,640 examples extracted from `human-eval.jsonl`. We parsed each example to extract function docstrings and surrounding comments (mostly Python), then used an OpenRouter "fast-grok" free model as a preprocessing / normalization step to:
   - split function-level docstrings into candidate comment sentences,
   - canonicalize parameter and return descriptions,
   - filter out non-descriptive or placeholder docstrings (e.g., single-word comments).

   The normalized candidates were programmatically labeled using heuristics (presence of function-level description, param/return mentions, and overlap with code identifiers). These heuristics produce silver-standard labels (higher noise than human annotation but useful signal).

We then combined the C-based Original training set with Python-derived silver-standard and rule-based synthetic examples to form the **Combined** training set (8,826 samples). Table 1 summarizes composition.

Table 1
Dataset composition after cleaning.

Dataset	Samples	Useful	Not Useful
Original seed	11,452	7,063 (61.7%)	4,389 (38.3%)
Rule-based synthetic	500	235 (47.0%)	265 (53.0%)
HumanEval silver	1,640	970 (59.1%)	670 (40.9%)
Training (Original)	8,326	–	–
Training (Combined)	8,826	–	–

3.2. Preprocessing and labeling notes

Code and comment texts were concatenated with a separator token. Tokenization uses CodeBERT's fast BPE tokenizer with a maximum sequence length of 512. For silver-standard examples we applied two lightweight filters: (1) minimum token length (3 words) and (2) presence of at least one verb or function-related identifier, to reduce trivial or metadata-only docstrings. The silver-standard labeling heuristics prioritize precision (favoring conservative "useful" labels) but inevitably inject label noise; we therefore treat these examples as complementary rather than authoritative.

3.3. Models and training

We continued MLM from `microsoft/codebert-base-mlm` on the Original (C) training corpus prior to supervised fine-tuning (15% mask, batch size 16, lr 5e-5, max 7 epochs, AdamW). After DAPT we replace the MLM head with a classification layer (dropout 0.1) and fine-tune on the binary task (batch size 32, lr 2e-5, up to 10 epochs, linear warmup 10% steps). Baselines use TF–IDF features with: logistic regression (L2), linear SVM, multinomial NB, and random forest (100 trees). Baselines use 10-fold CV for hyperparameter tuning.

3.4. Evaluation

Metrics: accuracy, precision, recall, weighted F_1. Experimental configurations: (A) train Original → test Original; (B) train Combined → test Combined; (C) Original → Combined; (D) Combined → Original. Results are reported on held-out test splits not used during validation.

3.5. Why MLM trained on C still helps after mixing Python silver data

Although silver examples derive from Python docstrings, the MLM DAPT step was performed primarily on C code and comments (Original corpus). Two reasons explain why DAPT remained effective after adding Python silver examples for classification:

1. **Cross-language lexical overlap:** many identifier names, API-style phrases, and comment conventions (e.g., "returns", "params", "compute") are shared across languages; the C-trained MLM learns these patterns and applies them when encoding Python-derived comments.
2. **Representation robustness:** MLM adapts CodeBERT's token and contextual embeddings to code/comment distributions (punctuation, identifier tokens, inline code snippets). These representational improvements transfer to mixed-language fine-tuning because the downstream classifier primarily relies on semantic patterns rather than language-specific syntax.

Empirically, mixing Python silver data produced only a slight drop in peak F_1 (0.9193 → 0.9166), indicating that distributional mismatch is limited and quality/noise in silver labels is the larger contributor to performance change.

4. Results

4.1. CodeBERT+MLM performance

Table 2 reports CodeBERT+MLM across configurations. The model achieves peak in-distribution F_1=0.9193 (Part A). Augmentation produces a small change (Part B F_1=0.9166).

Table 2
CodeBERT+MLM performance (held-out tests).

Configuration	Accuracy	F_1	Precision
Original → Original	0.9196	0.9193	0.9193
Combined → Combined	0.9179	0.9166	0.9193
Original → Combined	0.9145	0.9141	0.9141
Combined → Original	0.9190	0.9176	0.9203

4.2. Baselines and dynamics

Table 3 shows TF–IDF baseline results. Logistic regression is the strongest baseline (F_1=0.7067 on Combined), well below CodeBERT+MLM. Figure 2 shows training dynamics for MLM and fine-tuning.

MLM reduced perplexity notably over 7 epochs (from 15.3 to 8.7). Fine-tuning converged in 6–8 epochs with early stopping. Augmentation (+6% data) produced a 0.27pp F_1 decrease; manual inspection indicates silver-standard noise and stylistic mismatch as main causes.

4.3. Comparison with previous IRSE / FIRE submissions

We compare our best performance against published IRSE-relevant works (table below uses provided summary metadata). This contextualizes gains from DAPT+MLM.

Our CodeBERT+MLM result (F_1=0.9193) exceeds many classical/ML+LLM-augmented submissions while remaining below reported very large LLM one-off results (T7-9/T7-10) that use generative LMs as primary models. The comparison suggests DAPT is a practical and effective middle ground: much

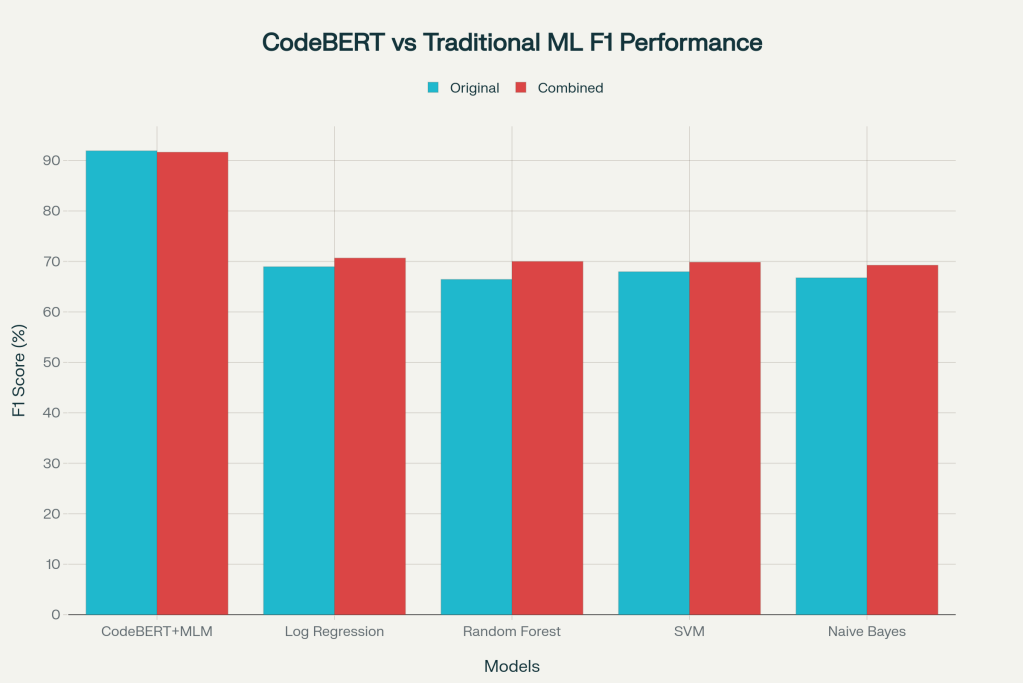

Figure 1: Performance comparison overview: CodeBERT+MLM vs TF–IDF baselines.

Table 3
TF–IDF baselines (accuracy / F_1).

Model	Dataset	Accuracy	F_1
Logistic Regression	Original	0.7203	0.6897
	Combined	0.7288	0.7067
SVM	Original	0.7041	0.6796
	Combined	0.7163	0.6985
Naive Bayes	Original	0.7023	0.6676
	Combined	0.7123	0.6925
Random Forest	Original	0.6771	0.6644
	Combined	0.7050	0.6999

Table 4
Selected previous IRSE/FIRE submissions (summary).

Paper ID	Title (short)	Primary model(s)	Best reported F_1
T7-1	Generative AI for Software Metadata (Majumdar et al.)	NN, SVM, RF	0.885
T7-2	ML-LLM pairing (Abi Akl)	NN, RF, Voting	0.884
T7-3	Software Metadata + GenAI	SVM, ANN	0.85 (approx)
T7-6	Enhancing Binary Comment Quality	BERT, DT, ANN	0.885
T7-9	Leveraging LMs for Code Comment	GPT-3 / Codex	0.96 (LLM results)
This work	CodeBERT + DAPT (MLM)	CodeBERT+MLM	**0.9193**

better than TF–IDF baselines and less resource-intensive than training or prompting very large LLMs end-to-end.

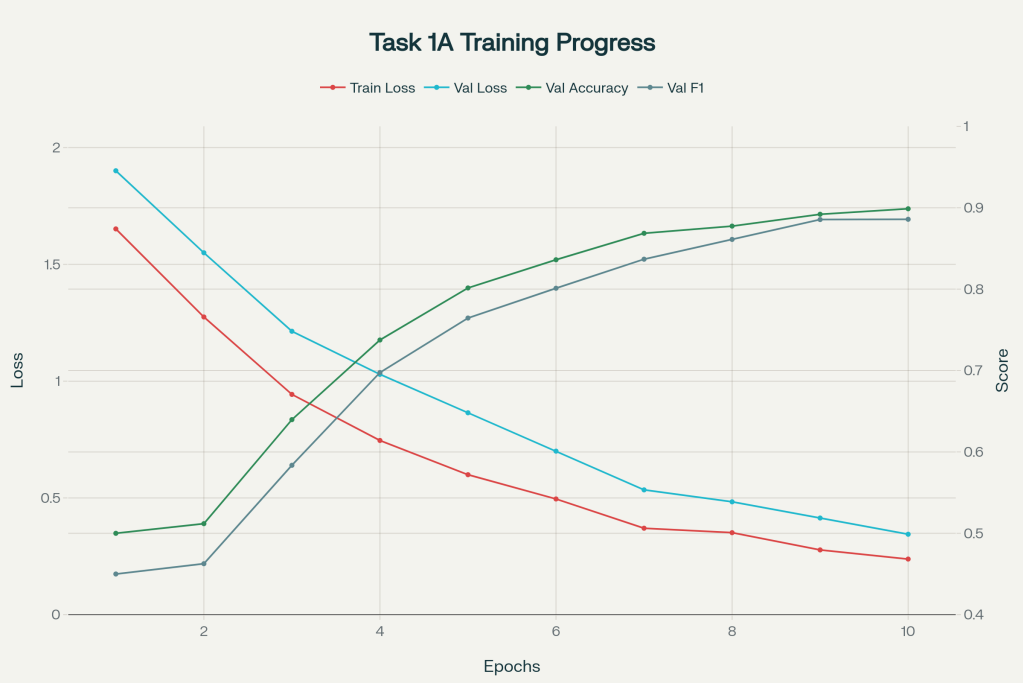

Figure 2: Training/validation trends (MLM + fine-tune): rapid MLM perplexity reduction and stable fine-tuning convergences.

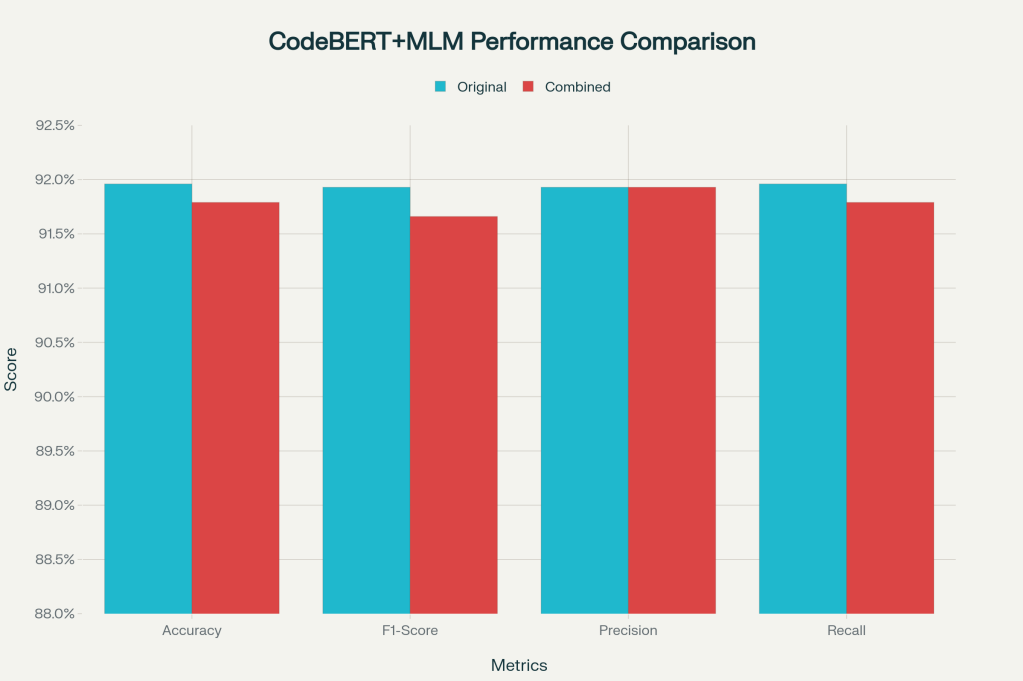

Figure 3: Final metric comparison (CodeBERT+MLM vs baselines).

5. Discussion

Domain-adaptive MLM significantly improves CodeBERT representations for comment relevance. Cross-evaluation shows training on Original and testing on Combined drops F_1 by only 0.52pp, indicating robust transfer. The small degradation with augmentation confirms that label quality and distribution alignment matter: silver-standard Python-derived examples add vocabulary and diversity, but their programmatic labels and stylistic differences introduce noise that slightly offsets gains.

Limitations: single-language (C) focus in DAPT, binary relevance label space, and the computational

cost for continued pretraining. Future work: multi-task DAPT, curriculum strategies that gradually introduce silver examples, active human verification of synthetic examples, and interpretability analyses to surface tokens driving decisions.

6. Conclusion

We show CodeBERT with domain-adaptive MLM attains strong results on comment relevance classification, outperforming TF–IDF baselines by a wide margin. Silver-standard augmentation from HumanEval (processed via OpenRouter fast-grok + heuristics) provides additional data but must be quality controlled to benefit high-capacity transformers. DAPT is a practical, resource-efficient strategy to improve specialized SE NLP tasks.

Declaration on Generative AI

In the course of preparing this manuscript, the author(s) employed the generative AI tool ChatGPT. Its use was limited to performing checks for grammar and spelling. Following this, the author(s) conducted a thorough review and revision of the text and assume full responsibility for the final published content.

References

[1] Rani, P., Panichella, S., Leuenberger, M., Ghafari, M., Nierstrasz, O.: A decade of code comment quality assessment: A systematic literature review. J. Syst. Softw. 196, 111515 (2023). https://doi.org/10.1016/j.jss.2022.111515

[2] Majumdar, S., Liang, Y., Ripley, M., Mody, N., Dalal, K., Chen, L.: Automated evaluation of comments to aid software maintenance. Empir. Softw. Eng. 27(5) (2022). https://doi.org/10.1007/s10664-022-10171-5

[3] Forum for Information Retrieval Evaluation (FIRE). https://fire.irsi.org.in (2024)

[4] Majumdar, S., Paul, S., Paul, D., Bandyopadhyay, A., Chattopadhyay, S., Das, P.P., Clough, P.D., Majumder, P.: Generative AI for software metadata: Overview of the Information Retrieval in Software Engineering Track at FIRE 2023. arXiv:2311.03374 (2023). https://arxiv.org/abs/2311.03374

[5] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., Zhou, M.: CodeBERT: A pre-trained model for programming and natural languages. In: Findings of EMNLP, pp. 1536–1547 (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.139

[6] Basu, T., Sruthi, S.: Identification of the relevance of comments in codes using bag of words and transformer based models. arXiv:2308.06144 (2023). https://arxiv.org/abs/2308.06144

[7] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., Smith, N.A.: Don't stop pretraining: Adapt language models to domains and tasks. In: Proceedings of ACL, pp. 8342–8360 (2020). https://doi.org/10.18653/v1/2020.acl-main.740

[8] Liu, K., Kim, D., Bissyandé, T.F., Yoo, S., Le Traon, Y.: Better exploiting CodeBERT to support source code-related tasks. In: ACM International Conference on Mining Software Repositories (MSR), pp. 545–549 (2022). https://doi.org/10.1145/3545258.3545260

[9] Mohammadkhani, A.H., Panichella, S., Parsa, S.: Explaining transformer-based code models: How do they learn code syntax and semantics? In: IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 519–523 (2022). https://doi.org/10.1109/ICSME58945.2022.00058

[10] Chen, X., et al.: Enhancing source code classification effectiveness via prompt engineering. Nat. Sci. Rep. 14, 69402 (2024). https://doi.org/10.1038/s41598-024-69402-7

[11] Kim, S., Lee, J., Park, M.: Subset selection for domain adaptive pre-training of language model. Nat. Sci. Rep. 15, 94085 (2025). https://doi.org/10.1038/s41598-025-94085-z

[12] Wang, Y., Zheng, Q., Chen, H., Yang, B., Xu, J., Gao, M.: Synthetic data generation with large language models for text classification. arXiv:2310.11467 (2023). https://arxiv.org/abs/2310.11467

[13] Feng, S.Y., et al.: A survey of data augmentation approaches for NLP. In: ACL Findings, pp. 968–988 (2021). https://doi.org/10.18653/v1/2021.findings-acl.84
[14] Majumdar, S., et al.: Overview of the Information Retrieval in Software Engineering Track at FIRE 2023. In: CEUR Workshop Proceedings, vol. 3681 (2023). https://ceur-ws.org/Vol-3681/T7-1.pdf
[15] Arumugam, R., Deborah, A.: Enhancing binary code comment quality classification: Integrating generative AI for improved accuracy. In: CEUR Workshop Proceedings, vol. 3681 (2023). https://ceur-ws.org/Vol-3681/T7-6.pdf
[16] Binary classification of source code comments using machine learning. In: CEUR Workshop Proceedings, vol. 3681 (2023). https://ceur-ws.org/Vol-3681/T7-7.pdf
[17] Majumdar, S., Deshpande, A., Das, P.P., Chakrabarti, P.P.: Comprehending C Codes with LLMs: Effective Comment Generation through Retrieval and Reasoning. Pattern Recognition Letters (2025). Elsevier.
[18] Paul, S., Majumdar, S., Shah, R., Das, S., Ghosh, M., Ganguly, D., Calikli, G., Sanyal, D., Das, P.P., Clough, P.D., et al.: Overview of the "Information Retrieval in Software Engineering"(IRSE) track at Forum for Information Retrieval 2024. In: Proceedings of the 16th Annual Meeting of the Forum for Information Retrieval Evaluation, pp. 18–21 (2024).
[19] Paul, S., Majumdar, S., Bandyopadhyay, A., Dave, B., Chattopadhyay, S., Das, P., Clough, P.D., Majumder, P.: Efficiency of Large Language Models to scale up Ground Truth: Overview of the IRSE Track at Forum for Information Retrieval 2023. In: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation, pp. 16–18 (2023).
[20] A. Deshpande, A. Maji, D. Mondol, P. P. Das, P. D. Clough, and S. Majumdar, "The Code–LLM Handshake: Smarter Maintenance Through AI," in *Proceedings of the 17th Annual Meeting of the Forum for Information Retrieval Evaluation*, pp. 9–12, 2025.
[21] A. Mitra, S. Majumdar, A. Mukhopadhyay, P. P. Das, P. D. Clough, and P. P. Chakrabarti, "Operationalizing Large Language Models with Design-Aware Contexts for Code Comment Generation," *arXiv preprint* arXiv:2510.22338, 2025.
[22] S. Majumdar and P. P. Das, "Smart Knowledge Transfer using Google-like Search," *arXiv preprint* arXiv:2308.06653, 2023.